FSx ONTAP para MLOps

NetApp artificial intelligence solutions

NetApp
February 12, 2026

Índice

FSx ONTAP para MLOps

Amazon FSx for NetApp ONTAP (FSx ONTAP) para MLOps

Esta seção se aprofunda na aplicação prática do desenvolvimento de infraestrutura de IA, fornecendo um passo a passo completo da construção de um pipeline MLOps usando o FSx ONTAP. Composto por três exemplos abrangentes, ele orienta você a atender às suas necessidades de MLOps por meio desta poderosa plataforma de gerenciamento de dados.

Esses artigos se concentram em:

1. "Parte 1 - Integrando o Amazon FSx for NetApp ONTAP (FSx ONTAP) como um bucket S3 privado no AWS SageMaker"
2. "Parte 2 - Aproveitando o Amazon FSx for NetApp ONTAP (FSx ONTAP) como uma fonte de dados para treinamento de modelos no SageMaker"
3. "Parte 3 - Construindo um Pipeline MLOps Simplificado (CI/CT/CD)"

Ao final desta seção, você terá adquirido uma sólida compreensão de como usar o FSx ONTAP para otimizar os processos de MLOps.

Parte 1 - Integrando o Amazon FSx for NetApp ONTAP (FSx ONTAP) como um bucket S3 privado no AWS SageMaker

Esta seção fornece um guia sobre como configurar o FSx ONTAP como um bucket S3 privado usando o AWS SageMaker.

Introdução

Usando o SageMaker como exemplo, esta página fornece orientação sobre como configurar o FSx ONTAP como um bucket S3 privado.

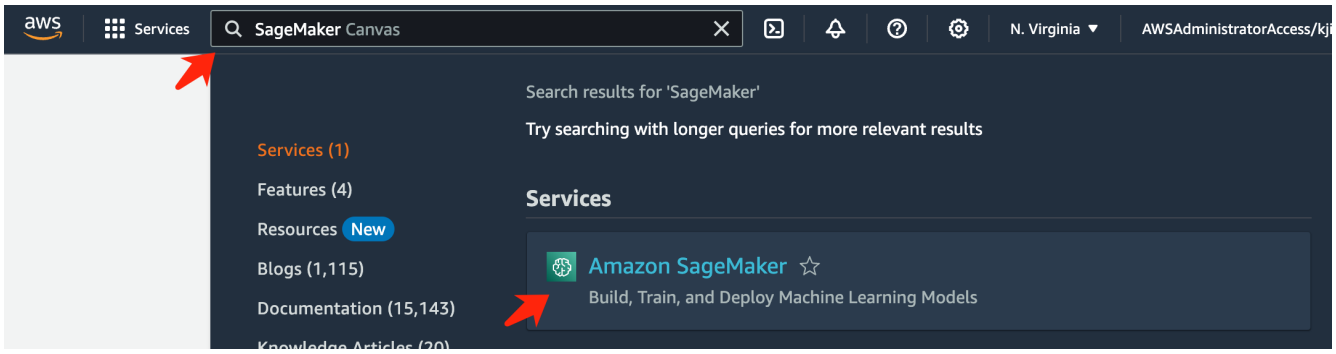
Para mais informações sobre o FSx ONTAP, consulte esta apresentação ("Link do vídeo")

Guia do usuário

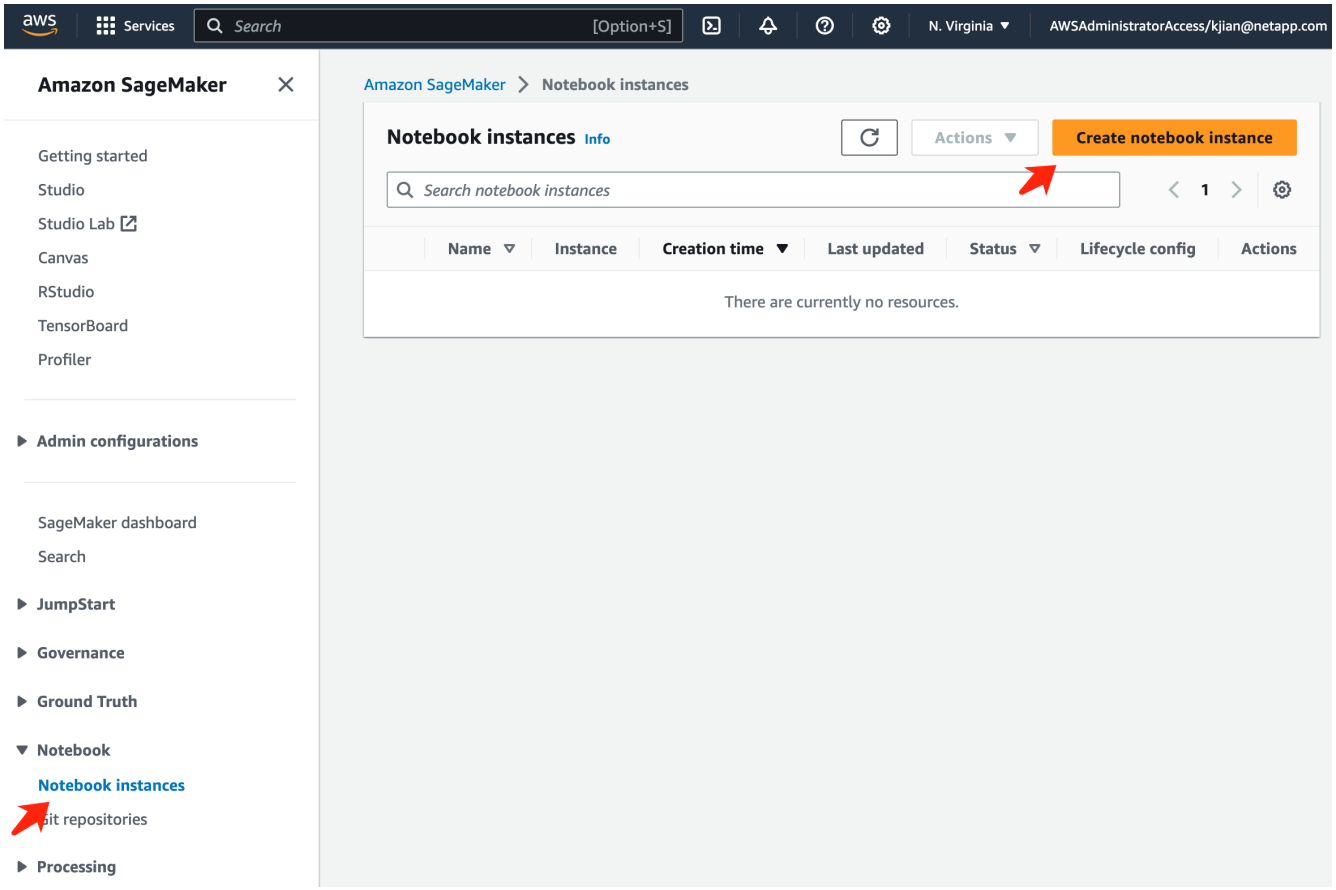
Criação de servidor

Criar uma instância do SageMaker Notebook

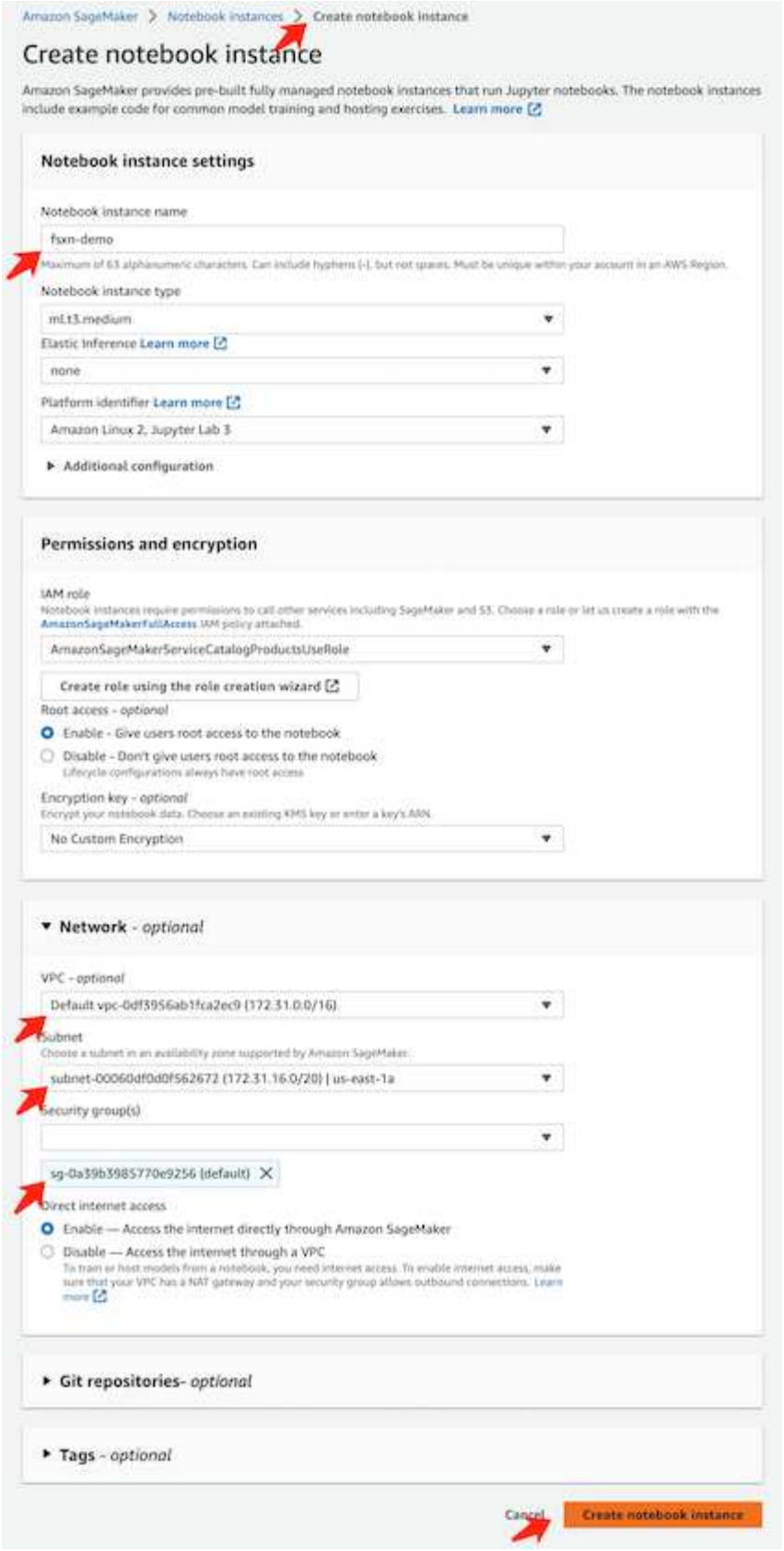
1. Abra o console da AWS. No painel de pesquisa, pesquise SageMaker e clique no serviço **Amazon SageMaker**.



2. Abra as **Instâncias do Notebook** na aba Notebook, clique no botão laranja **Criar instância do notebook**.

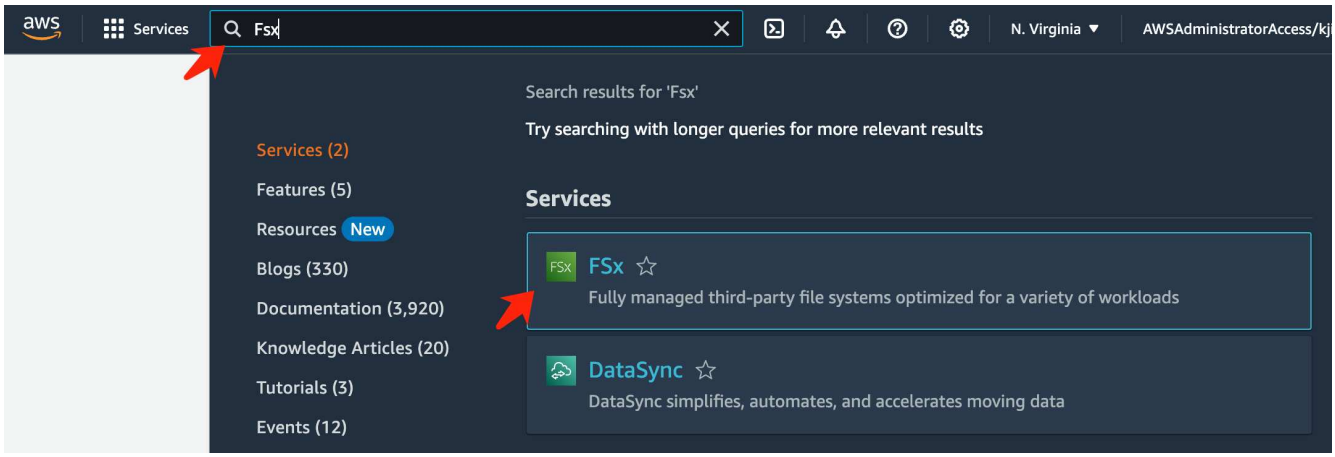


3. Na página de criação, insira o **Nome da instância do Notebook**. Expanda o painel **Rede**. Deixe as outras entradas padrão e selecione uma **VPC**, **Sub-rede** e **Grupo(s) de segurança**. (Esta **VPC** e **Sub-rede** serão usadas para criar o sistema de arquivos FSx ONTAP posteriormente) Clique no botão laranja **Criar instância de notebook** no canto inferior direito.

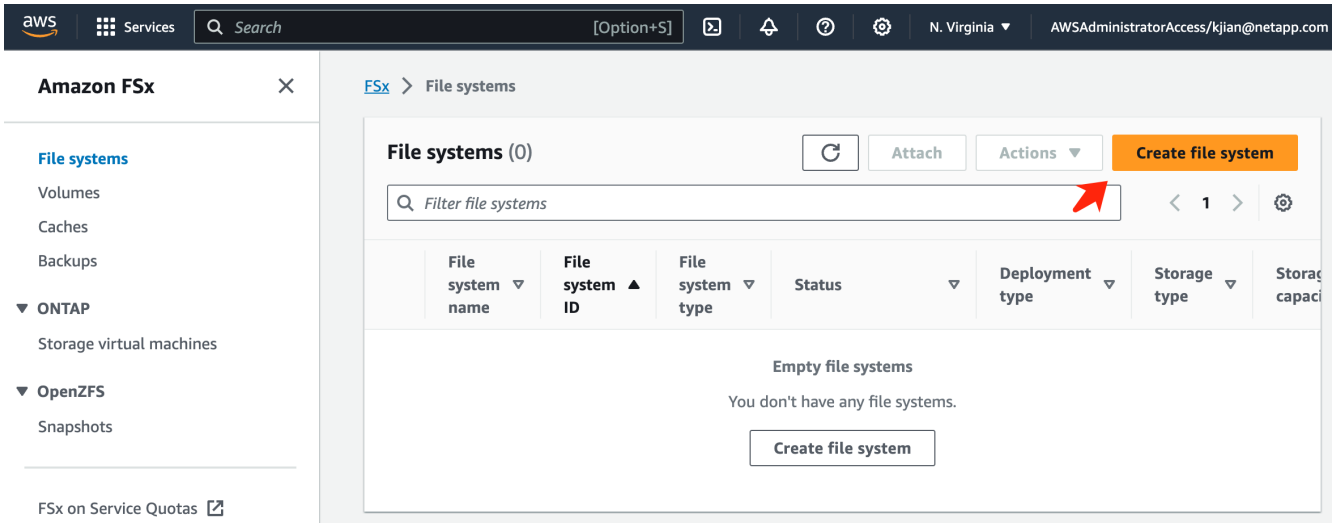


Crie um sistema de arquivos FSx ONTAP

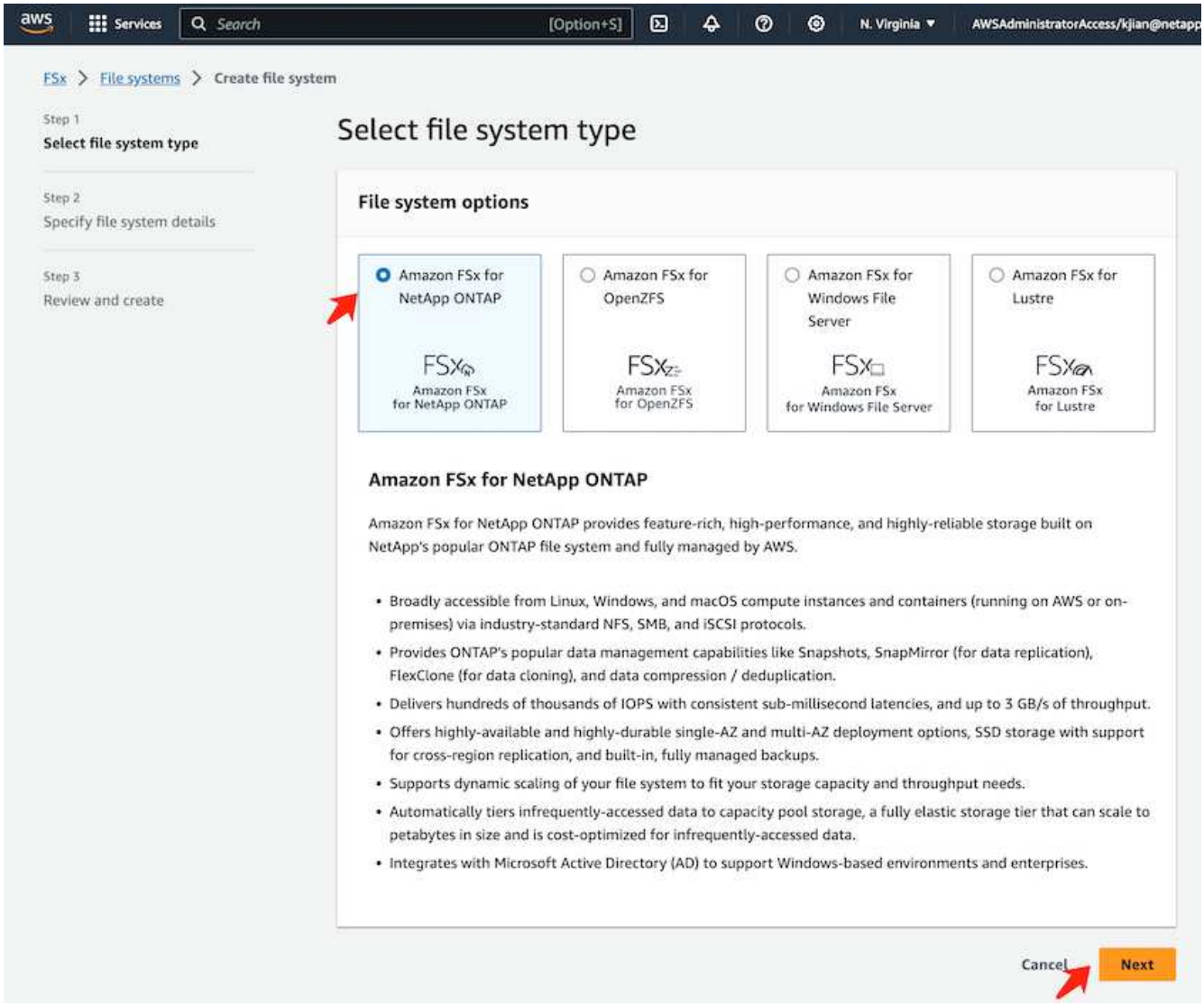
1. Abra o console da AWS. No painel de pesquisa, pesquise Fsx e clique no serviço **FSx**.



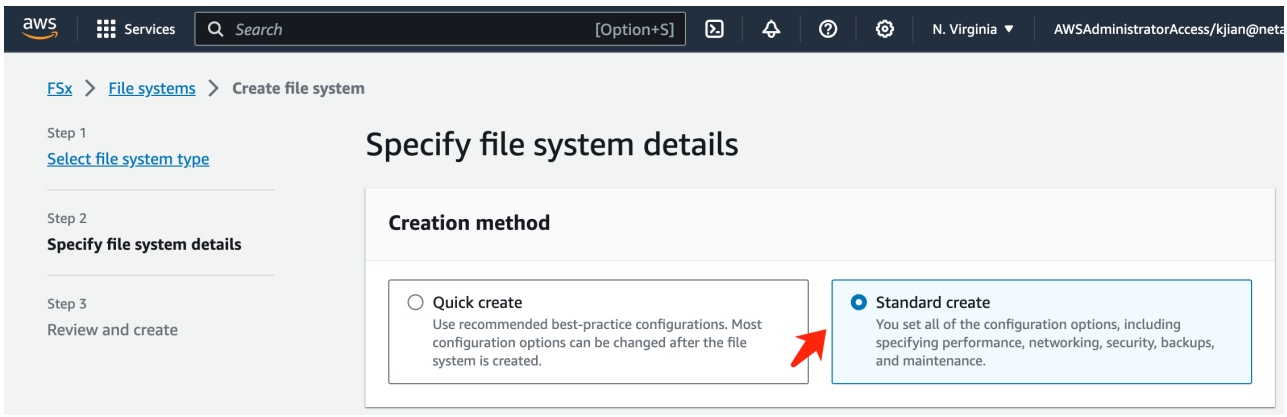
2. Clique em **Criar sistema de arquivos**.



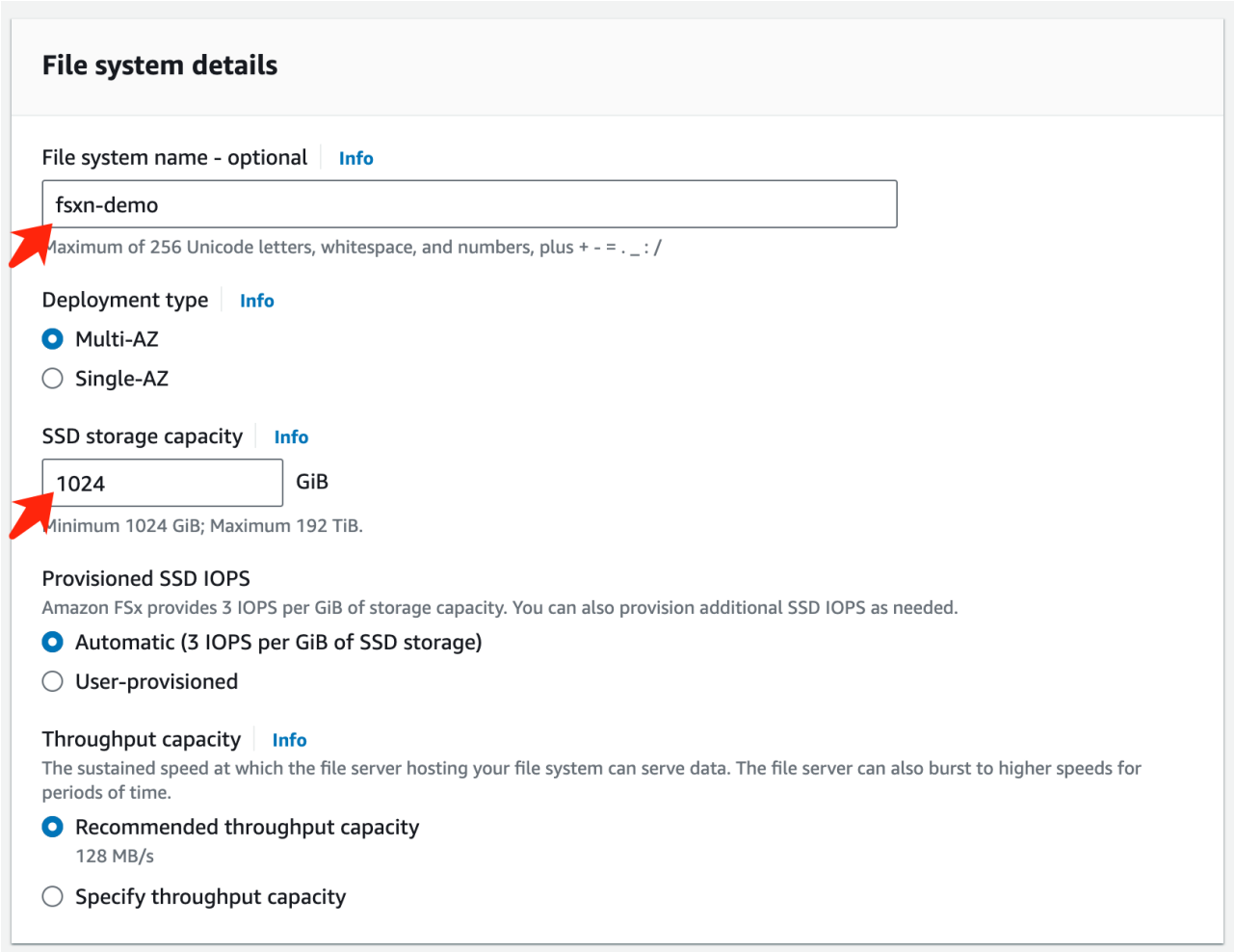
3. Selecione o primeiro cartão **FSx ONTAP** e clique em **Avançar**.



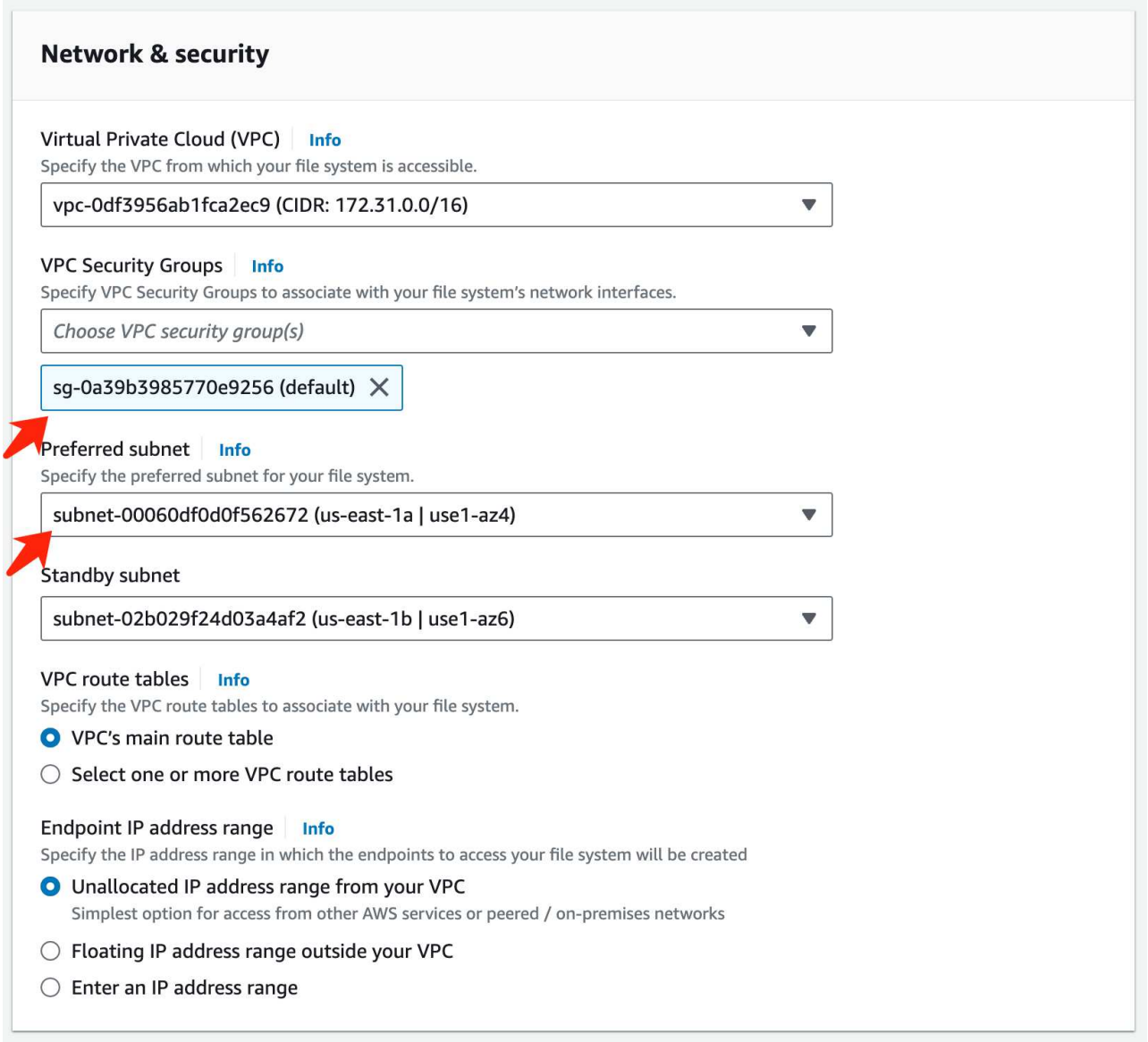
4. Na página de configuração de detalhes.
   a. Selecione a opção **Criação padrão**.



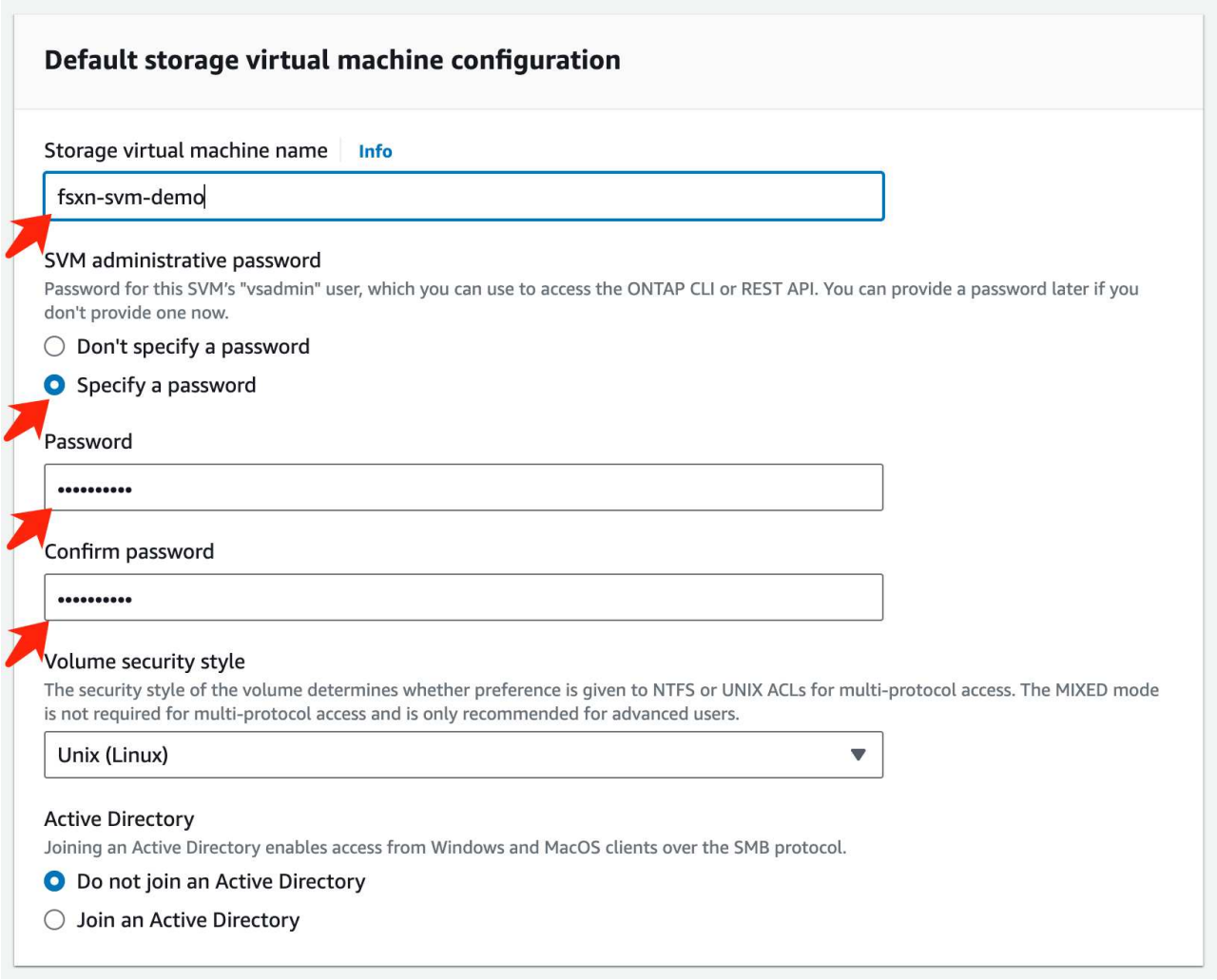
   b. Digite o **Nome do sistema de arquivos** e a **Capacidade de armazenamento do SSD**.



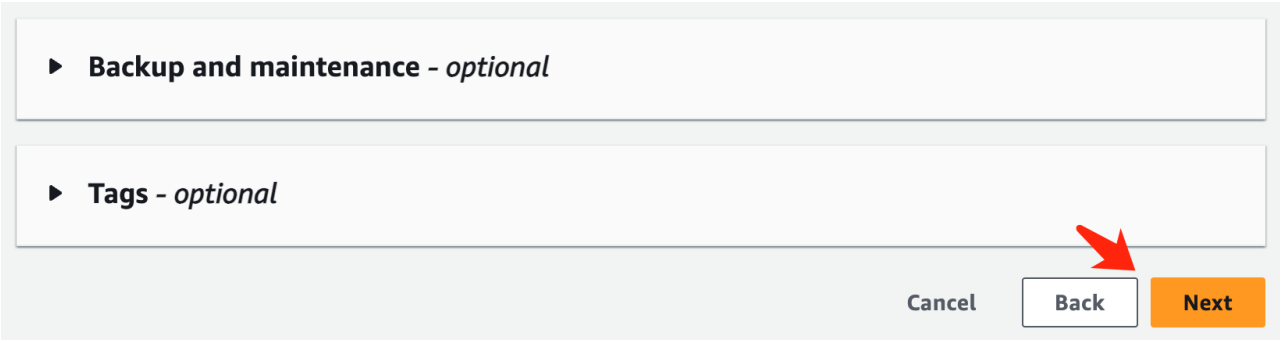
c. Certifique-se de usar a **VPC** e a **sub-rede** da mesma forma que a instância do **SageMaker Notebook**.



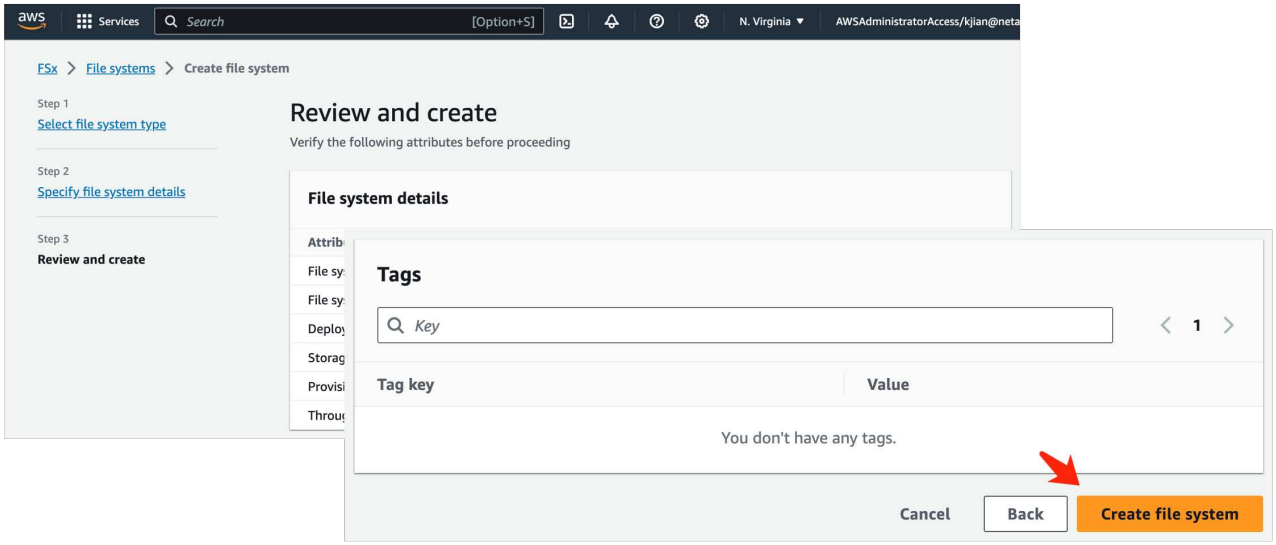
d. Digite o nome da **máquina virtual de armazenamento** e **especifique uma senha** para sua SVM (máquina virtual de armazenamento).



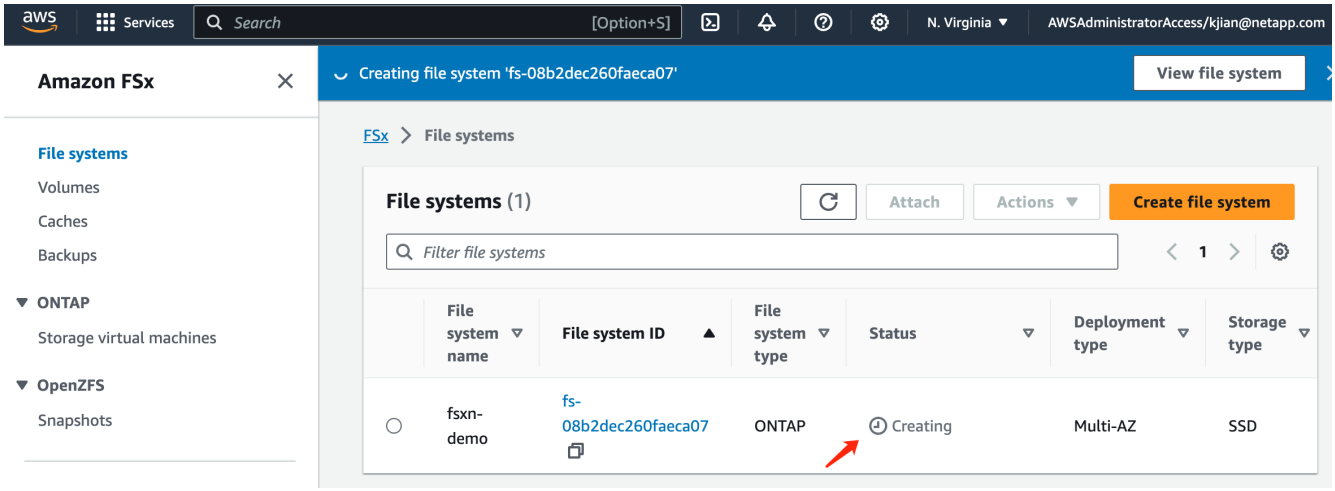
e. Deixe as outras entradas como padrão e clique no botão laranja **Avançar** no canto inferior direito.



f. Clique no botão laranja **Criar sistema de arquivos** no canto inferior direito da página de revisão.



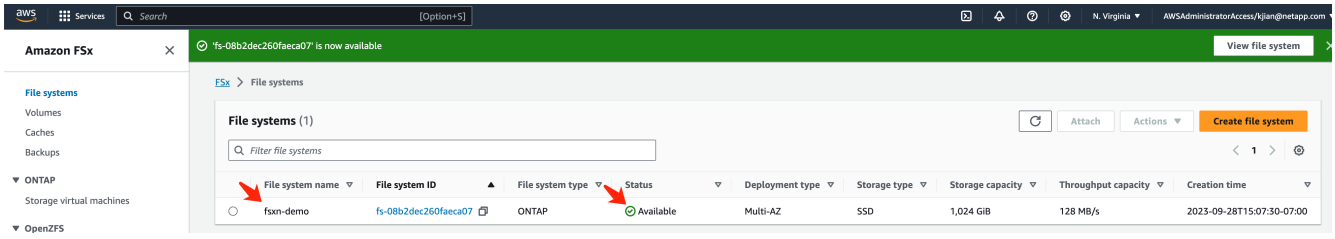
5. Pode levar cerca de **20-40 minutos** para iniciar o sistema de arquivos FSx.



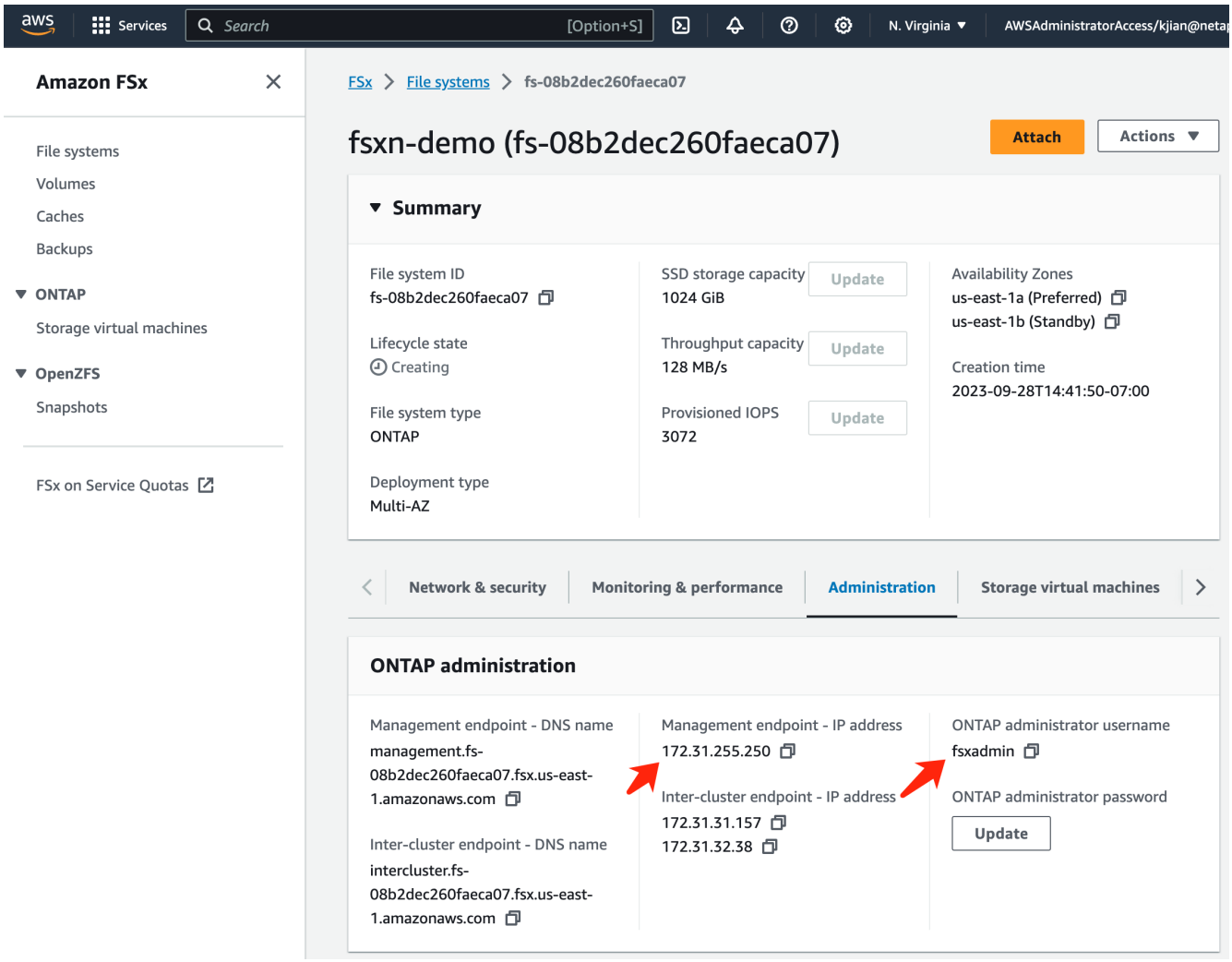
Configuração do servidor

Configuração ONTAP

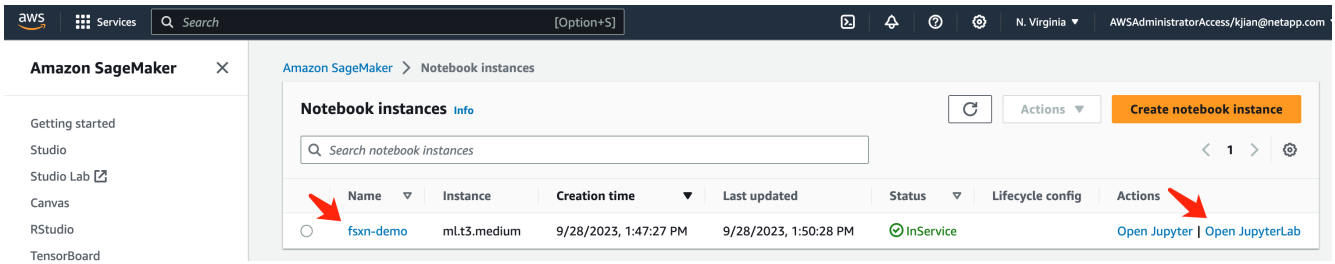
1. Abra o sistema de arquivos FSx criado. Certifique-se de que o status é **Disponível**.



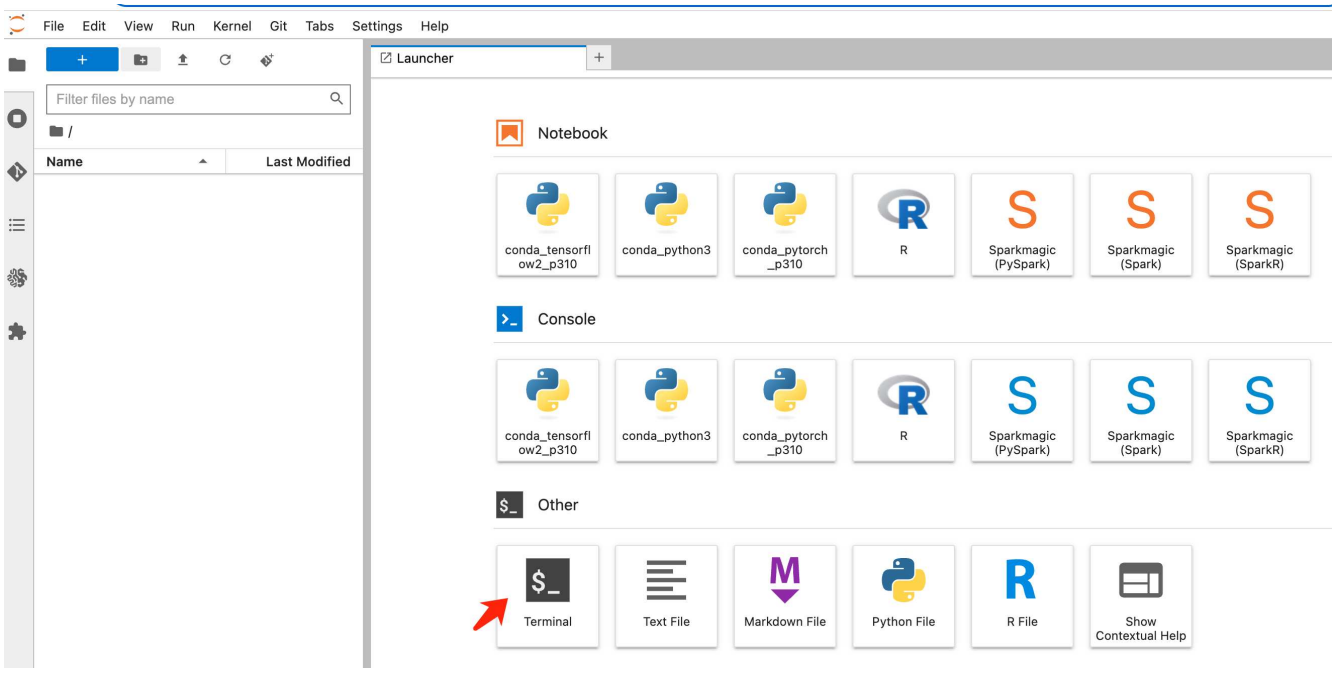
2. Selecione a aba **Administração** e mantenha o **Ponto de extremidade de gerenciamento - Endereço IP** e o *Nome de usuário do administrador do ONTAP *.



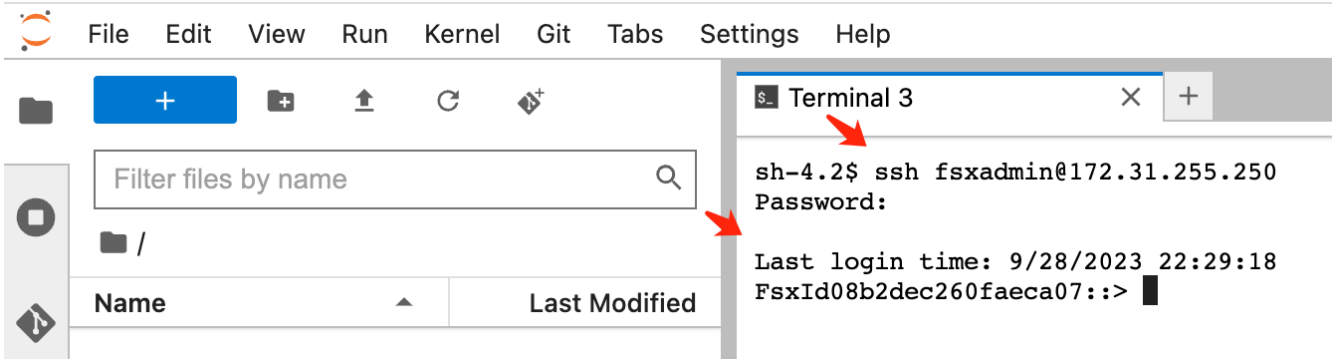
3. Abra a **instância do SageMaker Notebook** criada e clique em **Abrir JupyterLab**.



4. Na página do Jupyter Lab, abra um novo **Terminal**.



5. Digite o comando ssh ssh <nome de usuário administrador>@< IP do servidor ONTAP > para efetuar login no sistema de arquivos FSx ONTAP . (O nome de usuário e o endereço IP são recuperados na etapa 2) Use a senha usada ao criar a **Máquina virtual de armazenamento**.



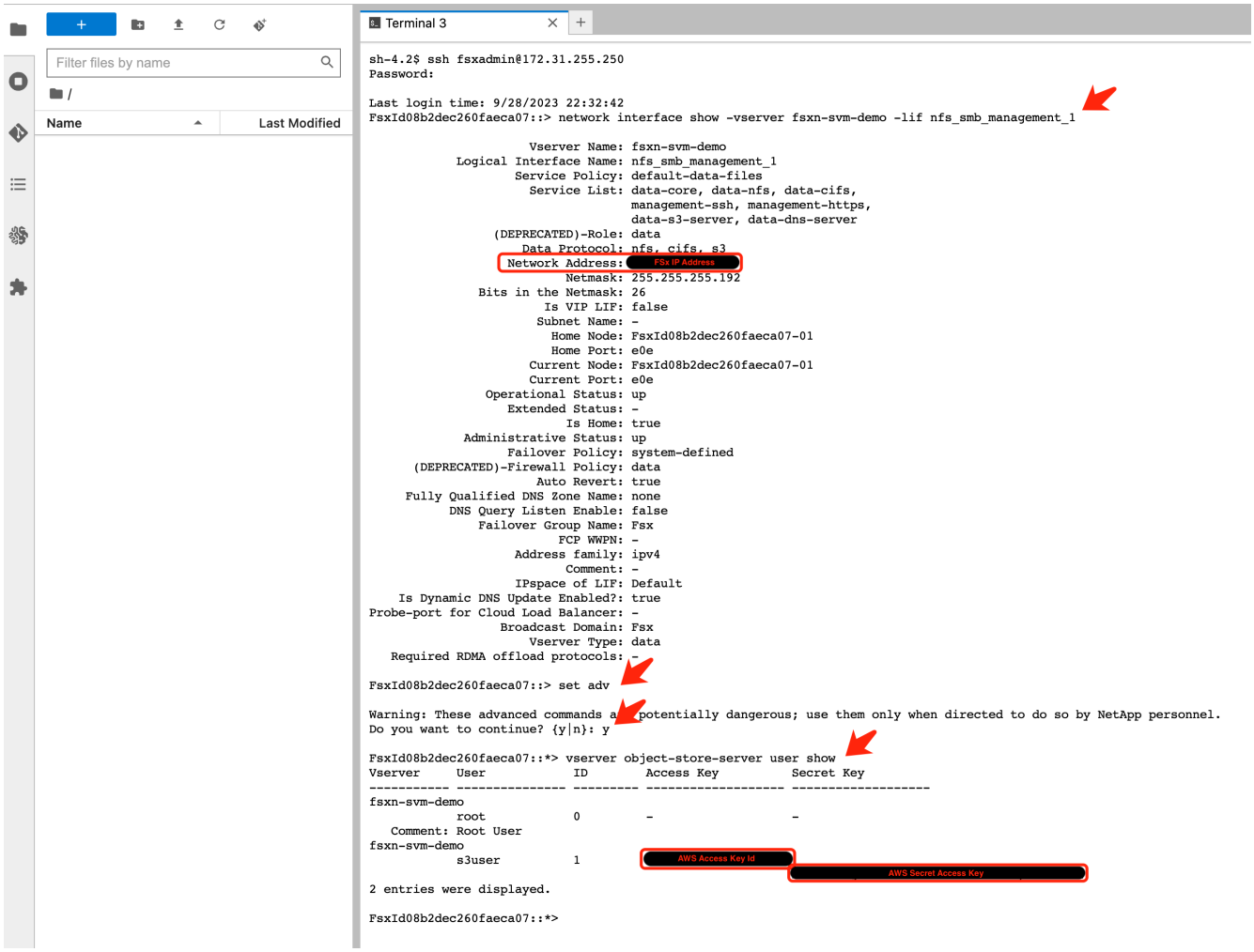
6. Execute os comandos na seguinte ordem. Usamos **fsxn-ontap** como nome para o **nome do bucket S3 privado do FSx ONTAP *. Use o *nome da máquina virtual de armazenamento** para o argumento **-vserver**.

```
vserver object-store-server create -vserver fsxn-svm-demo -object-store
-server fsx_s3 -is-http-enabled true -is-https-enabled false

vserver object-store-server user create -vserver fsxn-svm-demo -user
s3user

vserver object-store-server group create -name s3group -users s3user
-policies FullAccess

vserver object-store-server bucket create fsxn-ontap -vserver fsxn-svm-
demo -type nas -nas-path /vol1
```

7. Execute os comandos abaixo para recuperar o IP do endpoint e as credenciais do FSx ONTAP private S3.

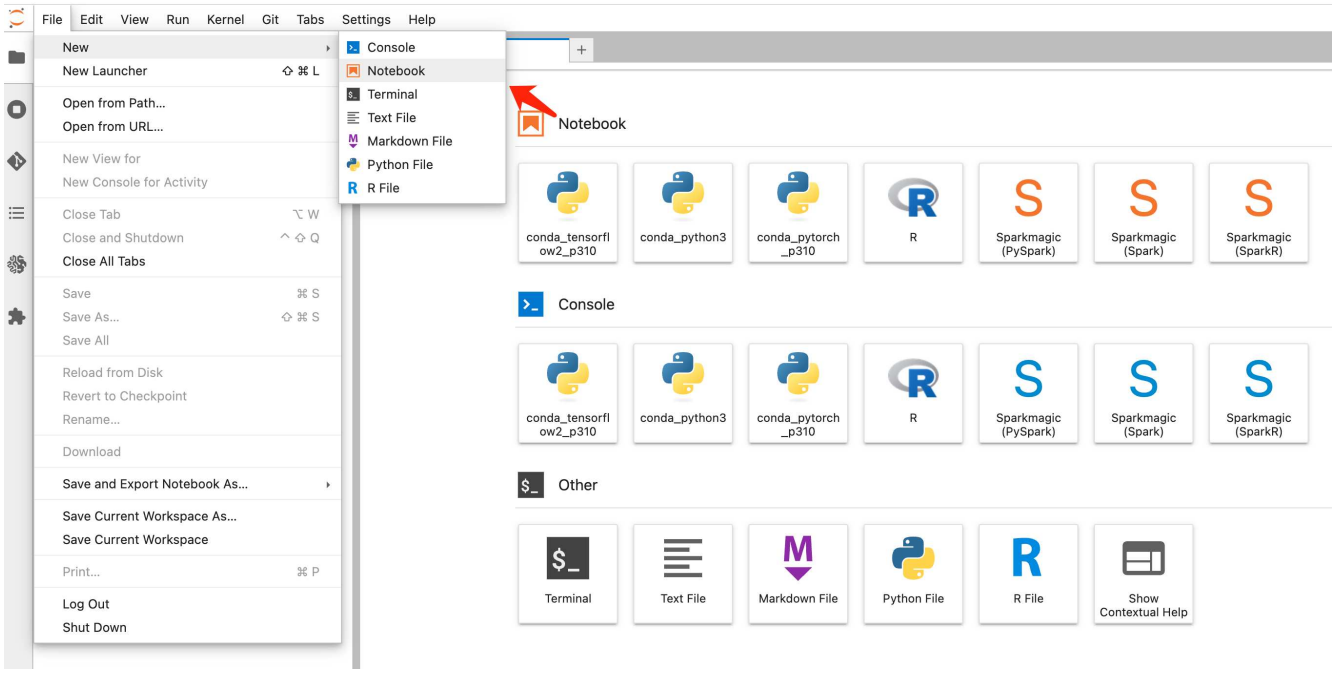
```
network interface show -vserver fsxn-svm-demo -lif nfs_smb_management_1

set adv

vserver object-store-server user show
```

8. Guarde o IP do endpoint e a credencial para uso futuro.



Configuração do cliente

1. Na instância do SageMaker Notebook, crie um novo notebook Jupyter.



2. Use o código abaixo como uma solução alternativa para carregar arquivos no bucket S3 privado do FSx ONTAP . Para um exemplo de código abrangente, consulte este notebook."fsxn_demo.ipynb"

```
# Setup configurations
# -------- Manual configurations --------
seed: int = 77                                          # Random
seed
bucket_name: str = 'fsxn-ontap'                         # The bucket
name in ONTAP
aws_access_key_id = '<Your ONTAP bucket key id>'        # Please get
this credential from ONTAP
aws_secret_access_key = '<Your ONTAP bucket access key>'    # Please get
this credential from ONTAP
fsx_endpoint_ip: str = '<Your FSx ONTAP IP address>'    # Please get
this IP address from FSx ONTAP
# -------- Manual configurations --------

# Workaround
## Permission patch
!mkdir -p vol1
!sudo mount -t nfs $fsx_endpoint_ip:/vol1 /home/ec2-user/SageMaker/vol1
!sudo chmod 777 /home/ec2-user/SageMaker/vol1

## Authentication for FSx ONTAP as a Private S3 Bucket
!aws configure set aws_access_key_id $aws_access_key_id
!aws configure set aws_secret_access_key $aws_secret_access_key
```

```
## Upload file to the FSx ONTAP Private S3 Bucket
%%capture
local_file_path: str = <Your local file path>

!aws s3 cp --endpoint-url http://$fsx_endpoint_ip /home/ec2-user
/SageMaker/$local_file_path  s3://$bucket_name/$local_file_path

# Read data from FSx ONTAP Private S3 bucket
## Initialize a s3 resource client
import boto3

# Get session info
region_name = boto3.session.Session().region_name

# Initialize Fsxn S3 bucket object
# --- Start integrating SageMaker with FSXN ---
# This is the only code change we need to incorporate SageMaker with
FSXN
s3_client: boto3.client = boto3.resource(
    's3',
    region_name=region_name,
    aws_access_key_id=aws_access_key_id,
    aws_secret_access_key=aws_secret_access_key,
    use_ssl=False,
    endpoint_url=f'http://{fsx_endpoint_ip}',
    config=boto3.session.Config(
        signature_version='s3v4',
        s3={'addressing_style': 'path'}
    )
)
# --- End integrating SageMaker with FSXN ---

## Read file byte content
bucket = s3_client.Bucket(bucket_name)

binary_data = bucket.Object(data.filename).get()['Body']
```

Isso conclui a integração entre o FSx ONTAP e a instância do SageMaker.

Lista de verificação de depuração útil

- Certifique-se de que a instância do SageMaker Notebook e o sistema de arquivos FSx ONTAP estejam na mesma VPC.
- Lembre-se de executar o comando **set dev** no ONTAP para definir o nível de privilégio como **dev**.

Perguntas frequentes (em 27 de setembro de 2023)

P: Por que estou recebendo o erro "**Ocorreu um erro (NotImplemented) ao chamar a operação CreateMultipartUpload: O comando s3 que você solicitou não foi implementado**" ao carregar arquivos no FSx ONTAP?

R: Como um bucket S3 privado, o FSx ONTAP suporta o upload de arquivos de até 100 MB. Ao usar o protocolo S3, arquivos maiores que 100 MB são divididos em pedaços de 100 MB, e a função 'CreateMultipartUpload' é chamada. Entretanto, a implementação atual do FSx ONTAP private S3 não suporta esta função.

P: Por que estou recebendo o erro "**Ocorreu um erro (AccessDenied) ao chamar as operações PutObject: Acesso negado**" ao carregar arquivos no FSx ONTAP?

R: Para acessar o bucket S3 privado do FSx ONTAP a partir de uma instância do SageMaker Notebook, troque as credenciais da AWS para as credenciais do FSx ONTAP . No entanto, conceder permissão de gravação à instância requer uma solução alternativa que envolve montar o bucket e executar o comando shell 'chmod' para alterar as permissões.

P: Como posso integrar o bucket S3 privado do FSx ONTAP com outros serviços do SageMaker ML?

R: Infelizmente, o SDK de serviços do SageMaker não fornece uma maneira de especificar o ponto de extremidade para o bucket privado do S3. Como resultado, o FSx ONTAP S3 não é compatível com serviços do SageMaker, como Sagemaker Data Wrangler, Sagemaker Clarify, Sagemaker Glue, Sagemaker Athena, Sagemaker AutoML e outros.

Parte 2 - Aproveitando o AWS Amazon FSx for NetApp ONTAP (FSx ONTAP) como uma fonte de dados para treinamento de modelos no SageMaker

Este artigo é um tutorial sobre como usar o Amazon FSx for NetApp ONTAP (FSx ONTAP) para treinar modelos PyTorch no SageMaker, especificamente para um projeto de classificação de qualidade de pneus.

Introdução

Este tutorial oferece um exemplo prático de um projeto de classificação de visão computacional, proporcionando experiência prática na construção de modelos de ML que utilizam o FSx ONTAP como fonte de dados no ambiente SageMaker. O projeto se concentra no uso do PyTorch, uma estrutura de aprendizado profundo, para classificar a qualidade dos pneus com base em imagens de pneus. Ele enfatiza o desenvolvimento de modelos de aprendizado de máquina usando o FSx ONTAP como fonte de dados no Amazon SageMaker.

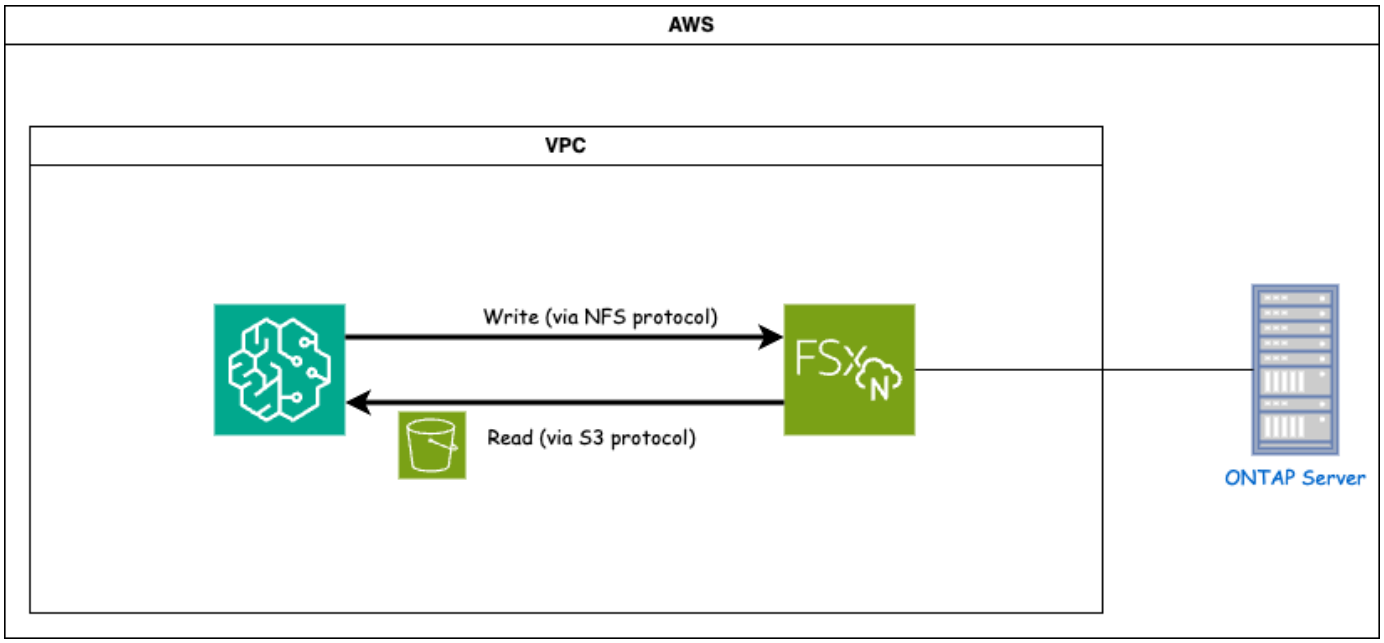
O que é FSx ONTAP

O Amazon FSx ONTAP é de fato uma solução de armazenamento totalmente gerenciada oferecida pela AWS. Ele aproveita o sistema de arquivos ONTAP da NetApp para fornecer armazenamento confiável e de alto desempenho. Com suporte para protocolos como NFS, SMB e iSCSI, ele permite acesso direto de diferentes instâncias de computação e contêineres. O serviço foi projetado para oferecer desempenho excepcional, garantindo operações de dados rápidas e eficientes. Ele também oferece alta disponibilidade e durabilidade, garantindo que seus dados permaneçam acessíveis e protegidos. Além disso, a capacidade de armazenamento do Amazon FSx ONTAP é escalável, permitindo que você a ajuste facilmente de acordo com

suas necessidades.

Pré-requisito

Ambiente de rede



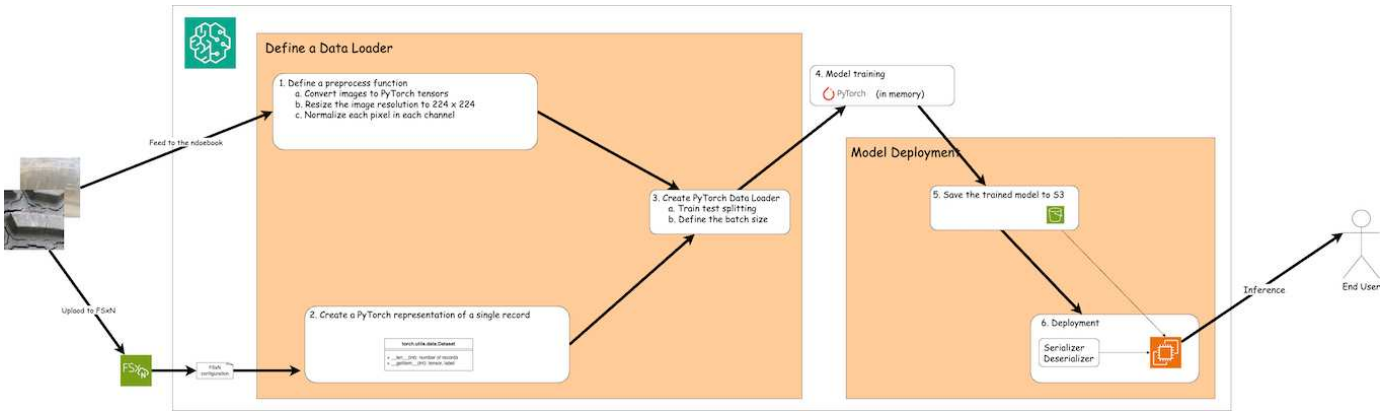
FSx ONTAP (Amazon FSx ONTAP) é um serviço de armazenamento da AWS. Ele inclui um sistema de arquivos em execução no sistema NetApp ONTAP e uma máquina virtual do sistema gerenciada pela AWS (SVM) que se conecta a ele. No diagrama fornecido, o servidor NetApp ONTAP gerenciado pela AWS está localizado fora da VPC. O SVM atua como intermediário entre o SageMaker e o sistema NetApp ONTAP , recebendo solicitações de operação do SageMaker e encaminhando-as para o armazenamento subjacente. Para acessar o FSx ONTAP, o SageMaker deve ser colocado na mesma VPC que a implantação do FSx ONTAP . Esta configuração garante a comunicação e o acesso aos dados entre o SageMaker e o FSx ONTAP.

Acesso a dados

Em cenários do mundo real, os cientistas de dados normalmente utilizam os dados existentes armazenados no FSx ONTAP para construir seus modelos de aprendizado de máquina. No entanto, para fins de demonstração, como o sistema de arquivos FSx ONTAP fica inicialmente vazio após a criação, é necessário carregar manualmente os dados de treinamento. Isso pode ser feito montando o FSx ONTAP como um volume no SageMaker. Depois que o sistema de arquivos for montado com sucesso, você poderá carregar seu conjunto de dados no local montado, tornando-o acessível para treinar seus modelos no ambiente SageMaker. Essa abordagem permite que você aproveite a capacidade de armazenamento e os recursos do FSx ONTAP enquanto trabalha com o SageMaker para desenvolvimento e treinamento de modelos.

O processo de leitura de dados envolve a configuração do FSx ONTAP como um bucket S3 privado. Para aprender as instruções detalhadas de configuração, consulte"Parte 1 - Integrando o Amazon FSx for NetApp ONTAP (FSx ONTAP) como um bucket S3 privado no AWS SageMaker"

Visão geral da integração



O fluxo de trabalho de uso de dados de treinamento no FSx ONTAP para criar um modelo de aprendizado profundo no SageMaker pode ser resumido em três etapas principais: definição do carregador de dados, treinamento do modelo e implantação. Em um nível alto, essas etapas formam a base de um pipeline de MLOps. No entanto, cada etapa envolve várias subetapas detalhadas para uma implementação abrangente. Essas subetapas abrangem várias tarefas, como pré-processamento de dados, divisão de conjuntos de dados, configuração de modelo, ajuste de hiperparâmetros, avaliação de modelo e implantação de modelo. Essas etapas garantem um processo completo e eficaz para criar e implantar modelos de aprendizado profundo usando dados de treinamento do FSx ONTAP no ambiente SageMaker.

Integração passo a passo

Loader de dados

Para treinar uma rede de aprendizado profundo PyTorch com dados, um carregador de dados é criado para facilitar a alimentação de dados. O carregador de dados não apenas define o tamanho do lote, mas também determina o procedimento de leitura e pré-processamento de cada registro dentro do lote. Ao configurar o carregador de dados, podemos lidar com o processamento de dados em lotes, permitindo o treinamento da rede de aprendizado profundo.

O carregador de dados consiste em 3 partes.

Função de pré-processamento

```
from torchvision import transforms

preprocess = transforms.Compose([
    transforms.ToTensor(),
    transforms.Resize((224,224)),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225]
    )
])
```

O trecho de código acima demonstra a definição de transformações de pré-processamento de imagem usando o módulo **torchvision.transforms**. Neste tutorial, o objeto de pré-processo é criado para aplicar uma série de transformações. Primeiro, a transformação **ToTensor()** converte a imagem em uma representação tensorial. Posteriormente, a transformação **Resize 224,224** redimensiona a imagem para um tamanho fixo de

224x224 pixels. Por fim, a transformação **Normalize()** normaliza os valores do tensor subtraindo a média e dividindo pelo desvio padrão ao longo de cada canal. Os valores de média e desvio padrão usados para normalização são comumente empregados em modelos de redes neurais pré-treinados. No geral, esse código prepara os dados da imagem para processamento posterior ou entrada em um modelo pré-treinado, convertendo-os em um tensor, redimensionando-os e normalizando os valores de pixels.

A classe de conjunto de dados PyTorch

```
import torch
from io import BytesIO
from PIL import Image


class FSxNImageDataset(torch.utils.data.Dataset):
    def __init__(self, bucket, prefix='', preprocess=None):
        self.image_keys = [
            s3_obj.key
            for s3_obj in list(bucket.objects.filter(Prefix=prefix).all())
        ]
        self.preprocess = preprocess

    def __len__(self):
        return len(self.image_keys)

    def __getitem__(self, index):
        key = self.image_keys[index]
        response = bucket.Object(key)

        label = 1 if key[13:].startswith('defective') else 0

        image_bytes = response.get()['Body'].read()
        image = Image.open(BytesIO(image_bytes))
        if image.mode == 'L':
            image = image.convert('RGB')

        if self.preprocess is not None:
            image = self.preprocess(image)
        return image, label
```

Esta classe fornece funcionalidade para obter o número total de registros no conjunto de dados e define o método de leitura de dados para cada registro. Dentro da função ***getitem***, o código utiliza o objeto de bucket boto3 S3 para recuperar os dados binários do FSx ONTAP. O estilo de código para acessar dados do FSx ONTAP é semelhante à leitura de dados do Amazon S3. A explicação a seguir se aprofunda no processo de criação do objeto privado S3 **bucket**.

FSx ONTAP como um repositório S3 privado

```
seed = 77                                                    # Random seed
bucket_name = '<Your ONTAP bucket name>'                     # The bucket
name in ONTAP
aws_access_key_id = '<Your ONTAP bucket key id>'             # Please get
this credential from ONTAP
aws_secret_access_key = '<Your ONTAP bucket access key>'     # Please get
this credential from ONTAP
fsx_endpoint_ip = '<Your FSx ONTAP IP address>'                  # Please
get this IP address from FSXN
```

```
import boto3

# Get session info
region_name = boto3.session.Session().region_name

# Initialize Fsxn S3 bucket object
# --- Start integrating SageMaker with FSXN ---
# This is the only code change we need to incorporate SageMaker with FSXN
s3_client: boto3.client = boto3.resource(
    's3',
    region_name=region_name,
    aws_access_key_id=aws_access_key_id,
    aws_secret_access_key=aws_secret_access_key,
    use_ssl=False,
    endpoint_url=f'http://{fsx_endpoint_ip}',
    config=boto3.session.Config(
        signature_version='s3v4',
        s3={'addressing_style': 'path'}
    )
)
# s3_client = boto3.resource('s3')
bucket = s3_client.Bucket(bucket_name)
# --- End integrating SageMaker with FSXN ---
```

Para ler dados do FSx ONTAP no SageMaker, é criado um manipulador que aponta para o armazenamento do FSx ONTAP usando o protocolo S3. Isso permite que o FSx ONTAP seja tratado como um bucket S3 privado. A configuração do manipulador inclui a especificação do endereço IP do FSx ONTAP SVM, o nome do bucket e as credenciais necessárias. Para uma explicação abrangente sobre como obter esses itens de configuração, consulte o documento em"Parte 1 - Integrando o Amazon FSx for NetApp ONTAP (FSx ONTAP) como um bucket S3 privado no AWS SageMaker" .

No exemplo mencionado acima, o objeto bucket é usado para instanciar o objeto do conjunto de dados PyTorch. O objeto dataset será explicado com mais detalhes na seção subsequente.

O Loader de dados PyTorch

```
from torch.utils.data import DataLoader
torch.manual_seed(seed)

# 1. Hyperparameters
batch_size = 64

# 2. Preparing for the dataset
dataset = FSxNImageDataset(bucket, 'dataset/tyre', preprocess=preprocess)

train, test = torch.utils.data.random_split(dataset, [1500, 356])

data_loader = DataLoader(dataset, batch_size=batch_size, shuffle=True)
```

No exemplo fornecido, um tamanho de lote de 64 é especificado, indicando que cada lote conterá 64 registros. Combinando a classe **Dataset** do PyTorch, a função de pré-processamento e o tamanho do lote de treinamento, obtemos o carregador de dados para treinamento. Este carregador de dados facilita o processo de iteração pelo conjunto de dados em lotes durante a fase de treinamento.

Treinamento de modelo

```
from torch import nn


class TyreQualityClassifier(nn.Module):
    def __init__(self):
        super().__init__()
        self.model = nn.Sequential(
            nn.Conv2d(3,32,(3,3)),
            nn.ReLU(),
            nn.Conv2d(32,32,(3,3)),
            nn.ReLU(),
            nn.Conv2d(32,64,(3,3)),
            nn.ReLU(),
            nn.Flatten(),
            nn.Linear(64*(224-6)*(224-6),2)
        )
    def forward(self, x):
        return self.model(x)
```

```
import datetime

num_epochs = 2
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

model = TyreQualityClassifier()
fn_loss = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)


model.to(device)
for epoch in range(num_epochs):
    for idx, (X, y) in enumerate(data_loader):
        X = X.to(device)
        y = y.to(device)

        y_hat = model(X)

        loss = fn_loss(y_hat, y)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        current_time = datetime.datetime.now().strftime("%Y-%m-%d
%H:%M:%S")
        print(f"Current Time: {current_time} - Epoch [{epoch+1}/
{num_epochs}]- Batch [{idx + 1}] - Loss: {loss}", end='\r')
```

Este código implementa um processo de treinamento padrão do PyTorch. Ele define um modelo de rede neural chamado **TyreQualityClassifier** usando camadas convolucionais e uma camada linear para classificar a qualidade dos pneus. O loop de treinamento itera sobre lotes de dados, calcula a perda e atualiza os parâmetros do modelo usando retropropagação e otimização. Além disso, ele imprime a hora atual, época, lote e perda para fins de monitoramento.

Implantação do modelo

Implantação

```
import io
import os
import tarfile
import sagemaker

# 1. Save the PyTorch model to memory
buffer_model = io.BytesIO()
traced_model = torch.jit.script(model)
torch.jit.save(traced_model, buffer_model)

# 2. Upload to AWS S3
sagemaker_session = sagemaker.Session()
bucket_name_default = sagemaker_session.default_bucket()
model_name = f'tyre_quality_classifier.pth'

# 2.1. Zip PyTorch model into tar.gz file
buffer_zip = io.BytesIO()
with tarfile.open(fileobj=buffer_zip, mode="w:gz") as tar:
    # Add PyTorch pt file
    file_name = os.path.basename(model_name)
    file_name_with_extension = os.path.split(file_name)[-1]
    tarinfo = tarfile.TarInfo(file_name_with_extension)
    tarinfo.size = len(buffer_model.getbuffer())
    buffer_model.seek(0)
    tar.addfile(tarinfo, buffer_model)

# 2.2. Upload the tar.gz file to S3 bucket
buffer_zip.seek(0)
boto3.resource('s3') \
    .Bucket(bucket_name_default) \
    .Object(f'pytorch/{model_name}.tar.gz') \
    .put(Body=buffer_zip.getvalue())
```

O código salva o modelo PyTorch no **Amazon S3** porque o SageMaker exige que o modelo seja armazenado no S3 para implantação. Ao carregar o modelo no **Amazon S3**, ele se torna acessível ao SageMaker, permitindo a implantação e a inferência no modelo implantado.

```
import time
from sagemaker.pytorch import PyTorchModel
from sagemaker.predictor import Predictor
from sagemaker.serializers import IdentitySerializer
from sagemaker.deserializers import JSONDeserializer


class TyreQualitySerializer(IdentitySerializer):
```

```
    CONTENT_TYPE = 'application/x-torch'

    def serialize(self, data):
        transformed_image = preprocess(data)
        tensor_image = torch.Tensor(transformed_image)

        serialized_data = io.BytesIO()
        torch.save(tensor_image, serialized_data)
        serialized_data.seek(0)
        serialized_data = serialized_data.read()

        return serialized_data


class TyreQualityPredictor(Predictor):
    def __init__(self, endpoint_name, sagemaker_session):
        super().__init__(
            endpoint_name,
            sagemaker_session=sagemaker_session,
            serializer=TyreQualitySerializer(),
            deserializer=JSONDeserializer(),
        )

sagemaker_model = PyTorchModel(
    model_data=f's3://{bucket_name_default}/pytorch/{model_name}.tar.gz',
    role=sagemaker.get_execution_role(),
    framework_version='2.0.1',
    py_version='py310',
    predictor_cls=TyreQualityPredictor,
    entry_point='inference.py',
    source_dir='code',
)

timestamp = int(time.time())
pytorch_endpoint_name = '{}-{}-{}'.format('tyre-quality-classifier', 'pt',
timestamp)
sagemaker_predictor = sagemaker_model.deploy(
    initial_instance_count=1,
    instance_type='ml.p3.2xlarge',
    endpoint_name=pytorch_endpoint_name
)
```

Este código facilita a implantação de um modelo PyTorch no SageMaker. Ele define um serializador personalizado, **TyreQualitySerializer**, que pré-processa e serializa dados de entrada como um tensor PyTorch. A classe **TyreQualityPredictor** é um preditor personalizado que utiliza o serializador definido e um **JSONDeserializer**. O código também cria um objeto **PyTorchModel** para especificar o local S3 do modelo, a função do IAM, a versão do framework e o ponto de entrada para inferência. O código gera um registro de

data e hora e constrói um nome de ponto de extremidade com base no modelo e no registro de data e hora. Por fim, o modelo é implantado usando o método deploy, especificando a contagem de instâncias, o tipo de instância e o nome do endpoint gerado. Isso permite que o modelo PyTorch seja implantado e fique acessível para inferência no SageMaker.

Inferência

```
image_object = list(bucket.objects.filter('dataset/tyre'))[0].get()
image_bytes = image_object['Body'].read()

with Image.open(with Image.open(BytesIO(image_bytes)) as image:
    predicted_classes = sagemaker_predictor.predict(image)

    print(predicted_classes)
```

Este é o exemplo de uso do ponto de extremidade implantado para fazer a inferência.

Parte 3 - Construindo um Pipeline MLOps Simplificado (CI/CT/CD)

Este artigo fornece um guia para criar um pipeline de MLOps com serviços da AWS, com foco em retreinamento automatizado de modelos, implantação e otimização de custos.

Introdução

Neste tutorial, você aprenderá como aproveitar vários serviços da AWS para construir um pipeline de MLOps simples que abrange integração contínua (CI), treinamento contínuo (CT) e implantação contínua (CD). Ao contrário dos pipelines tradicionais de DevOps, o MLOps requer considerações adicionais para concluir o ciclo operacional. Ao seguir este tutorial, você obterá insights sobre como incorporar CT no loop MLOps, permitindo o treinamento contínuo de seus modelos e implantação perfeita para inferência. O tutorial guiará você pelo processo de utilização dos serviços da AWS para estabelecer esse pipeline de MLOps de ponta a ponta.

Manifesto

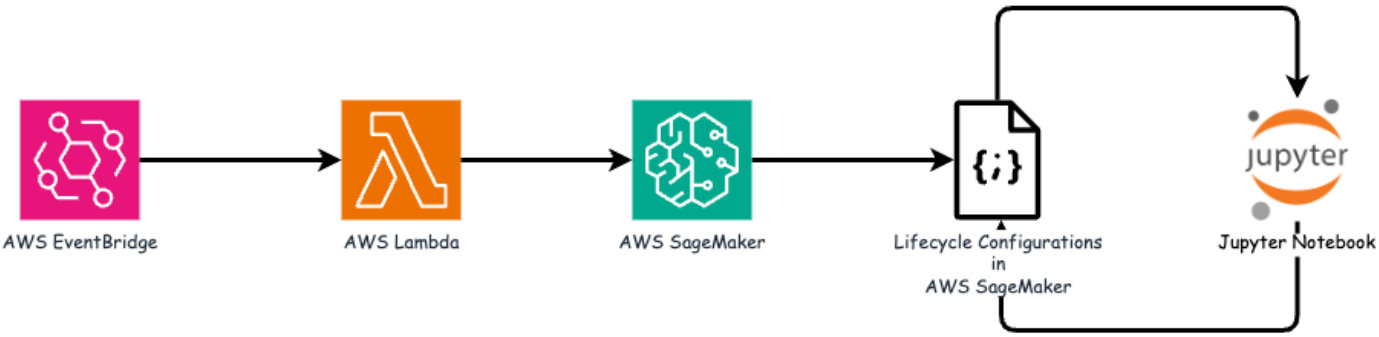
Funcionalidade	Nome	Comentário
Armazenamento de dados	AWS FSx ONTAP	Consulte "Parte 1 - Integrando o Amazon FSx for NetApp ONTAP (FSx ONTAP) como um bucket S3 privado no AWS SageMaker" .
IDE de ciência de dados	AWS SageMaker	Este tutorial é baseado no notebook Jupyter apresentado em"Parte 2 - Aproveitando o Amazon FSx for NetApp ONTAP (FSx ONTAP) como uma fonte de dados para treinamento de modelos no SageMaker" .

Funcionalidade	Nome	Comentário
Função para disparar o pipeline MLOps	Função AWS Lambda	-
Gatilho de tarefa cron	AWS EventBridge	-
Estrutura de aprendizagem profunda	PyTorch	-
SDK do AWS Python	boto3	-
Linguagem de programação	Pitão	v3.10

Pré-requisito

- Um sistema de arquivos FSx ONTAP pré-configurado. Este tutorial utiliza dados armazenados no FSx ONTAP para o processo de treinamento.
- Uma **instância do SageMaker Notebook** configurada para compartilhar a mesma VPC que o sistema de arquivos FSx ONTAP mencionado acima.
- Antes de acionar a **função AWS Lambda**, certifique-se de que a **instância do SageMaker Notebook** esteja no status **interrompido**.
- O tipo de instância **ml.g4dn.xlarge** é necessário para aproveitar a aceleração da GPU necessária para os cálculos de redes neurais profundas.

Arquitetura



Este pipeline MLOps é uma implementação prática que utiliza uma tarefa cron para acionar uma função sem servidor, que por sua vez executa um serviço da AWS registrado com uma função de retorno de chamada do ciclo de vida. O **AWS EventBridge** atua como uma tarefa cron. Ele invoca periodicamente uma **função AWS Lambda** responsável por retreinar e reimplantar o modelo. Este processo envolve a criação da instância do **AWS SageMaker Notebook** para executar as tarefas necessárias.

Configuração passo a passo

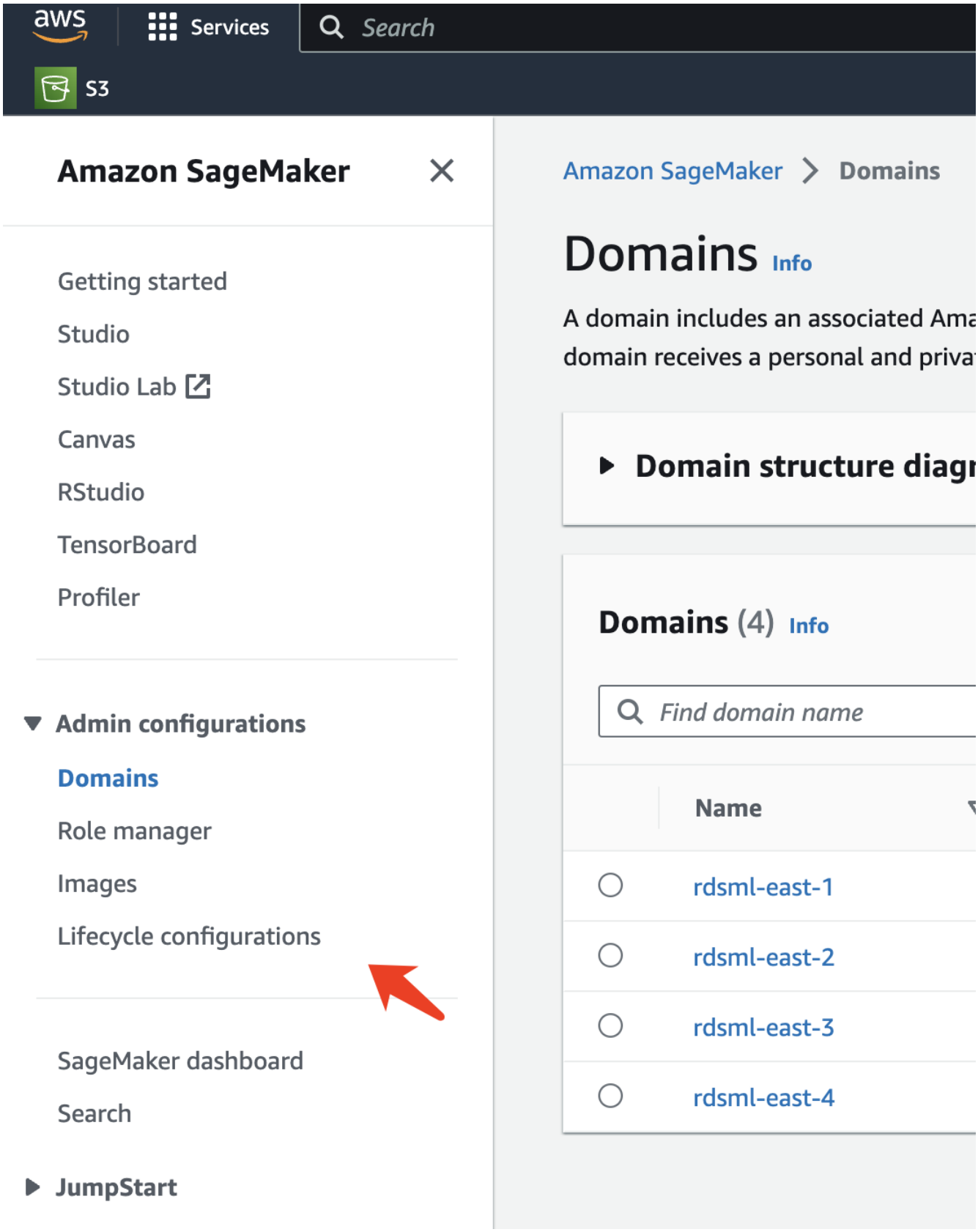
Configurações do ciclo de vida

Para configurar a função de retorno de chamada do ciclo de vida para a instância do AWS SageMaker Notebook, você utilizaria **Configurações do ciclo de vida**. Este serviço permite que você defina as ações necessárias a serem executadas durante a inicialização da instância do notebook. Especificamente, um script de shell pode ser implementado dentro das **Configurações do ciclo de vida** para desligar automaticamente a instância do notebook assim que os processos de treinamento e implantação forem concluídos. Esta é uma

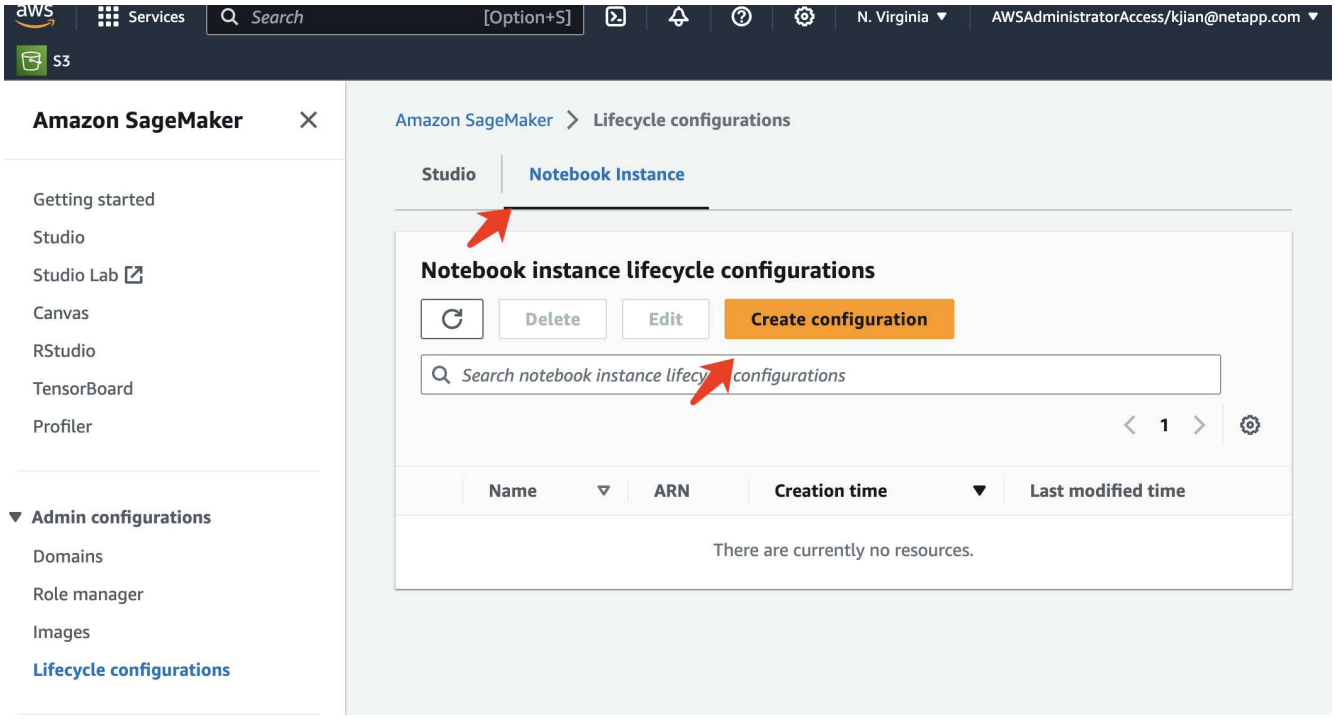
configuração necessária, pois o custo é uma das principais considerações em MLOps.

É importante observar que a configuração para **Configurações do ciclo de vida** precisa ser definida com antecedência. Portanto, é recomendável priorizar a configuração desse aspecto antes de prosseguir com a configuração de outros pipelines do MLOps.

1. Para definir as configurações do ciclo de vida, abra o painel **Sagemaker** e navegue até **Configurações do ciclo de vida** na seção **Configurações do administrador**.



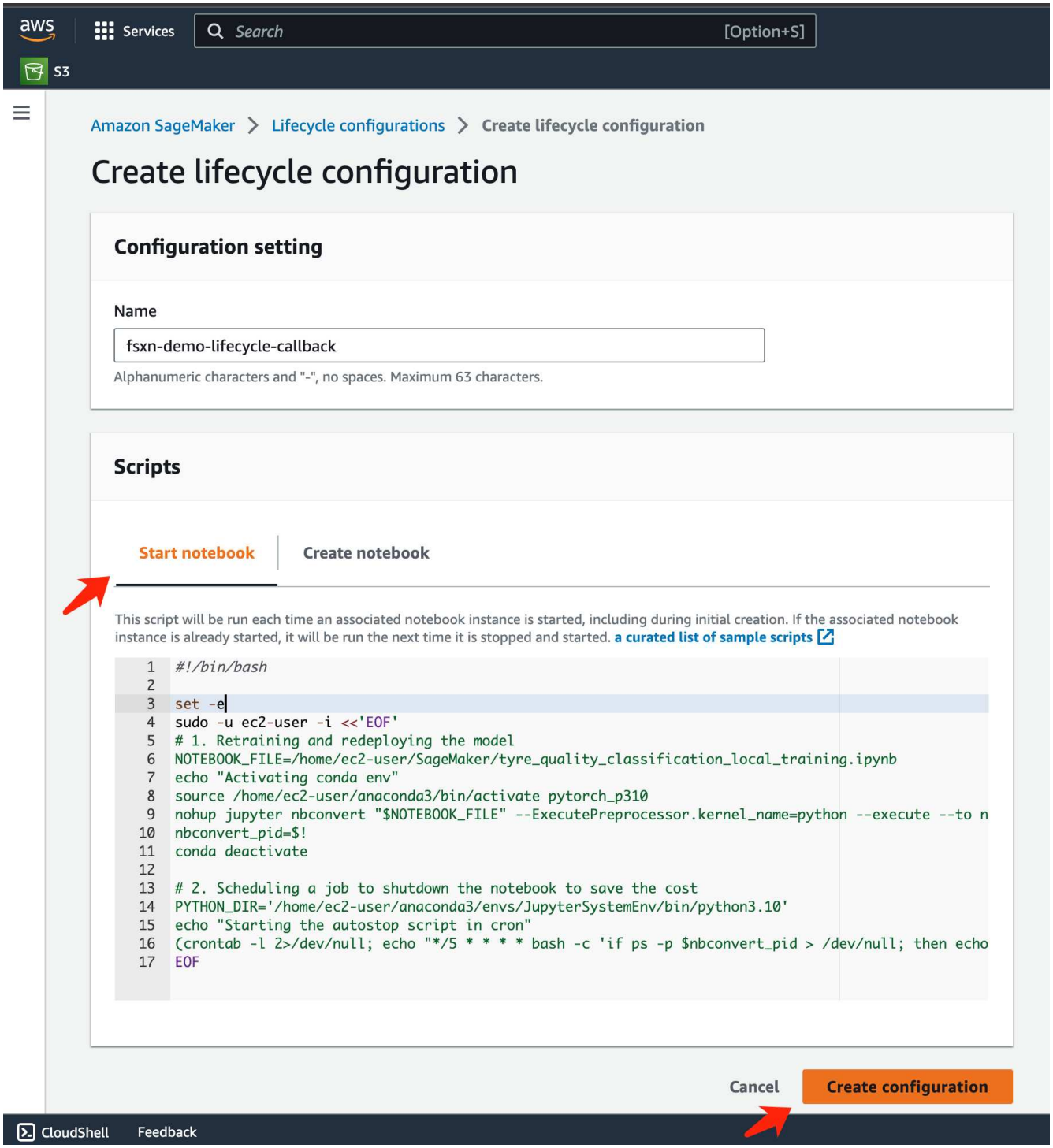
2. Selecione a aba **Instância do Notebook** e clique no botão **Criar configuração**



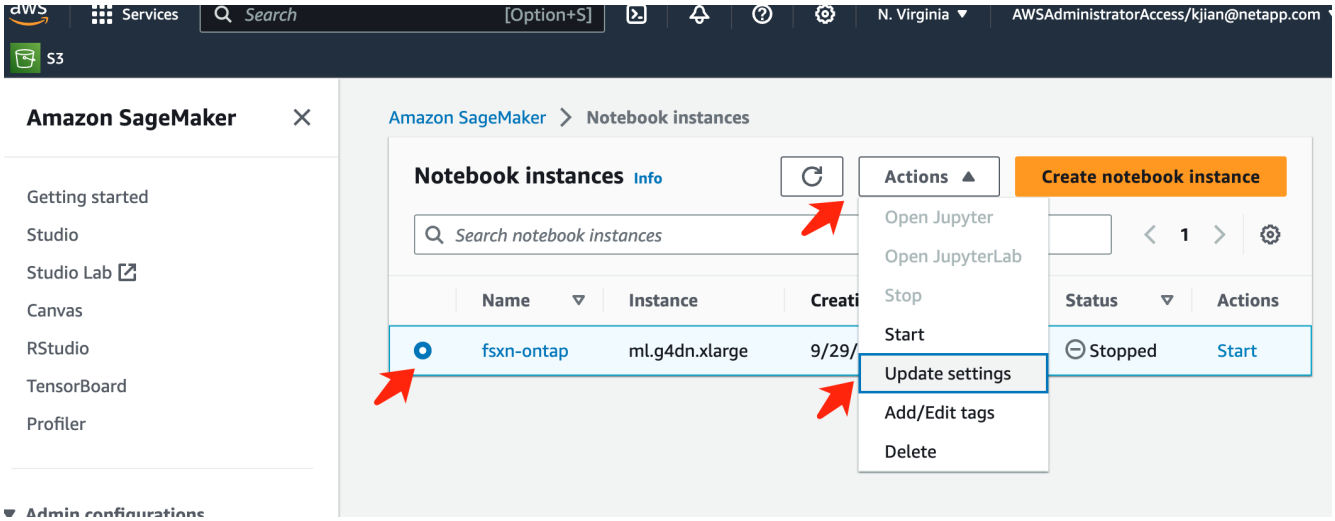
3. Cole o código abaixo na área de entrada.

```
#!/bin/bash

set -e
sudo -u ec2-user -i <<'EOF'
# 1. Retraining and redeploying the model
NOTEBOOK_FILE=/home/ec2-
user/SageMaker/tyre_quality_classification_local_training.ipynb
echo "Activating conda env"
source /home/ec2-user/anaconda3/bin/activate pytorch_p310
nohup jupyter nbconvert "$NOTEBOOK_FILE"
--ExecutePreprocessor.kernel_name=python --execute --to notebook &
nbconvert_pid=$!
conda deactivate

# 2. Scheduling a job to shutdown the notebook to save the cost
PYTHON_DIR='/home/ec2-
user/anaconda3/envs/JupyterSystemEnv/bin/python3.10'
echo "Starting the autostop script in cron"
(crontab -l 2>/dev/null; echo "*/5 * * * * bash -c 'if ps -p
$nbconvert_pid > /dev/null; then echo \"Notebook is still running.\" >>
/var/log/jupyter.log; else echo \"Notebook execution completed.\" >>
/var/log/jupyter.log; $PYTHON_DIR -c \"import boto3;boto3.client(
\'sagemaker\').stop_notebook_instance(NotebookInstanceName=get_notebook_
name())\" >> /var/log/jupyter.log; fi'") | crontab -
EOF
```

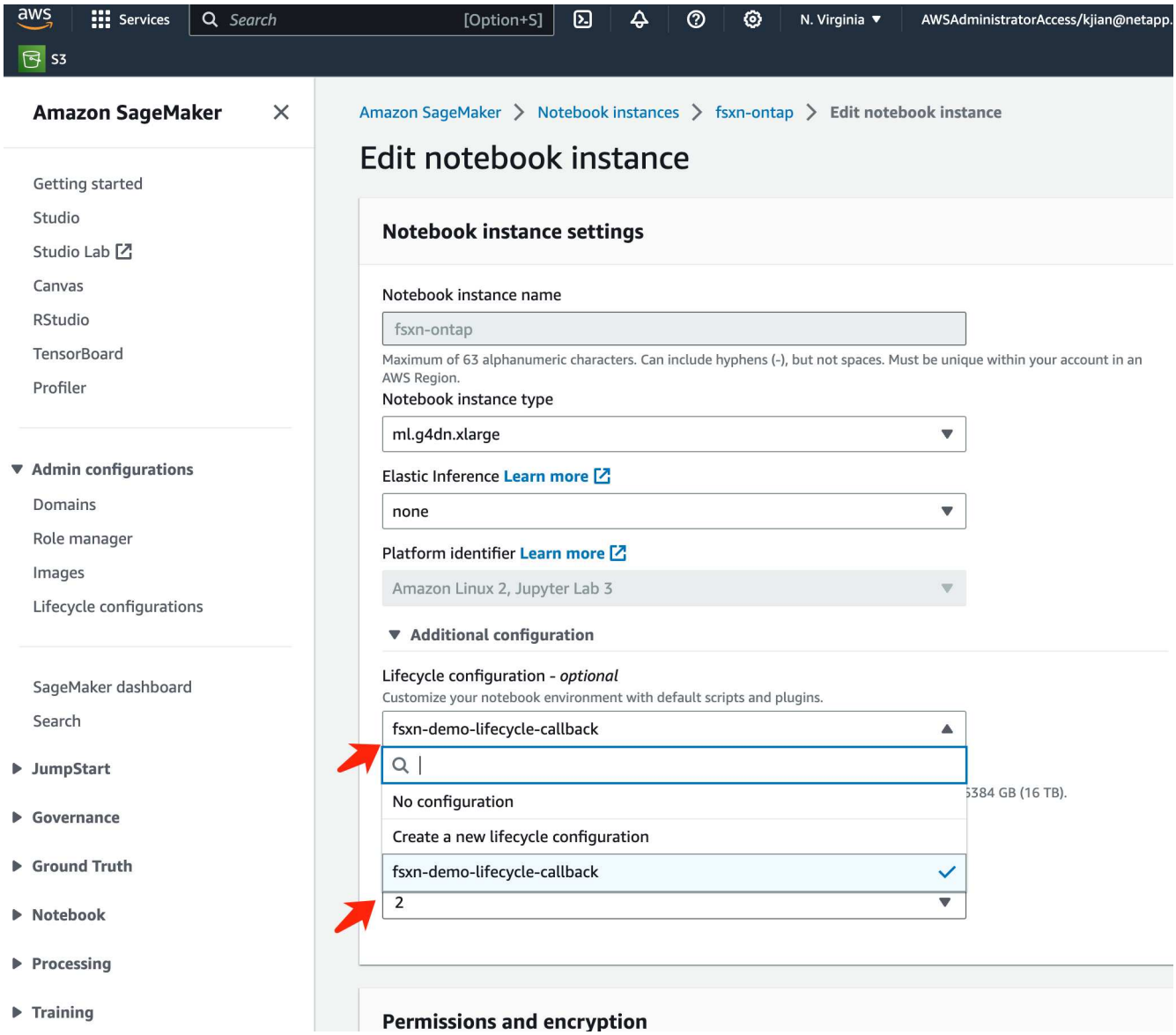
4. Este script executa o Jupyter Notebook, que lida com o retreinamento e a reimplantação do modelo para inferência. Após a conclusão da execução, o notebook será desligado automaticamente em 5 minutos. Para saber mais sobre a declaração do problema e a implementação do código, consulte"Parte 2 - Aproveitando o Amazon FSx for NetApp ONTAP (FSx ONTAP) como uma fonte de dados para treinamento de modelos no SageMaker" .



5. Após a criação, navegue até Instâncias do Notebook, selecione a instância de destino e clique em **Atualizar configurações** no menu suspenso Ações.



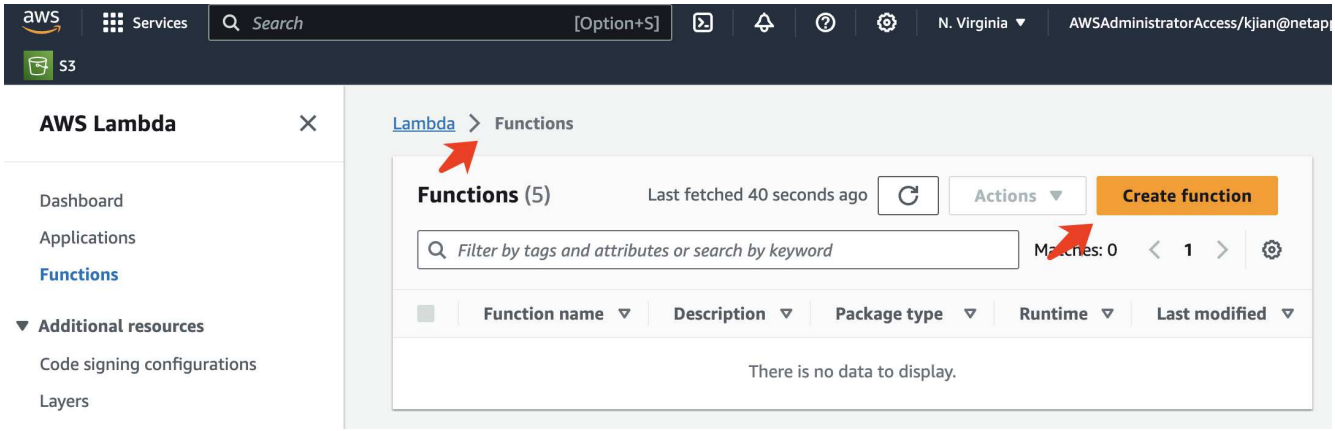
6. Selecione a **Configuração do ciclo de vida** criada e clique em **Atualizar instância do notebook**.



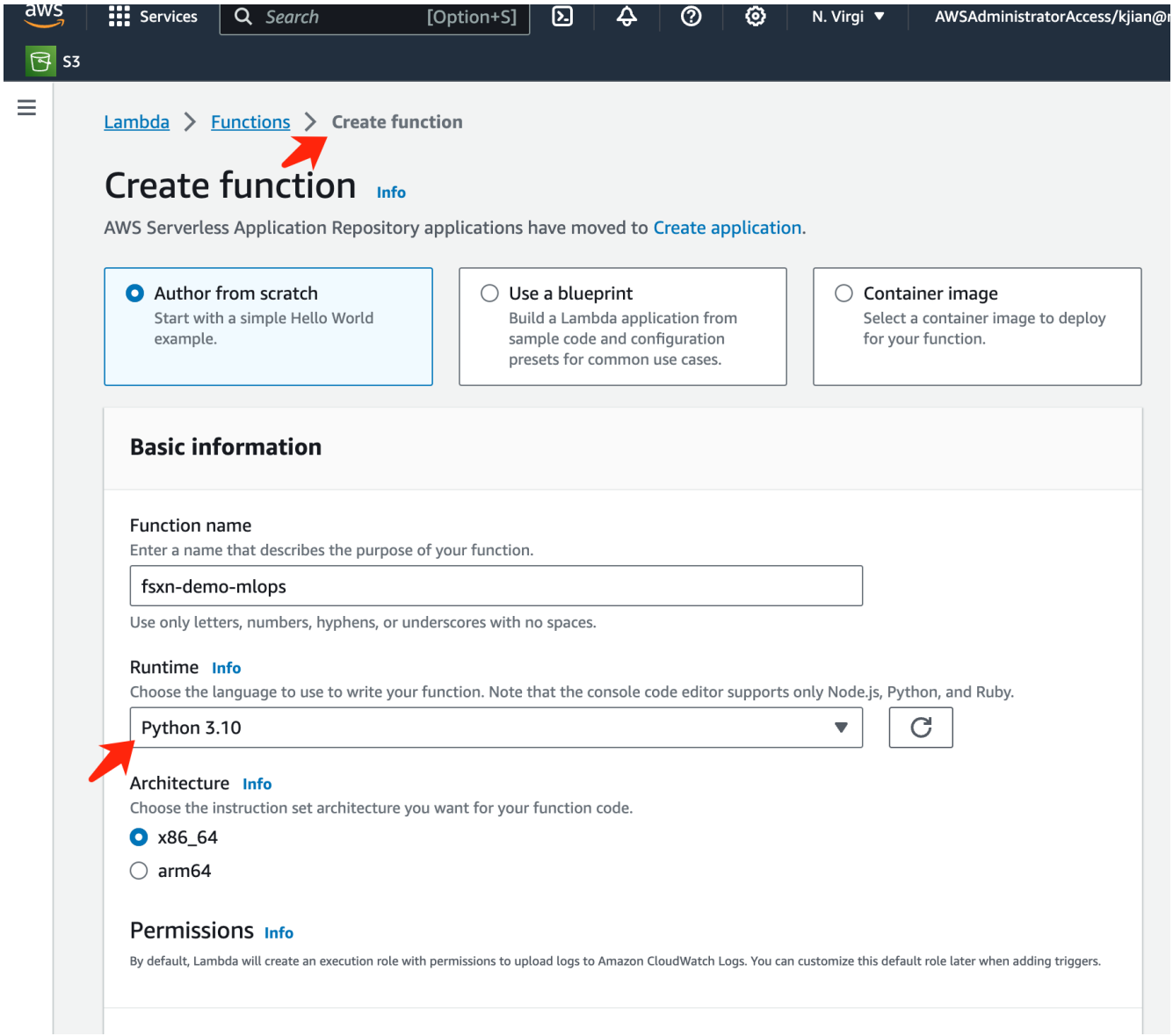
Função sem servidor do AWS Lambda

Como mencionado anteriormente, a **função AWS Lambda** é responsável por iniciar a **instância do AWS SageMaker Notebook**.

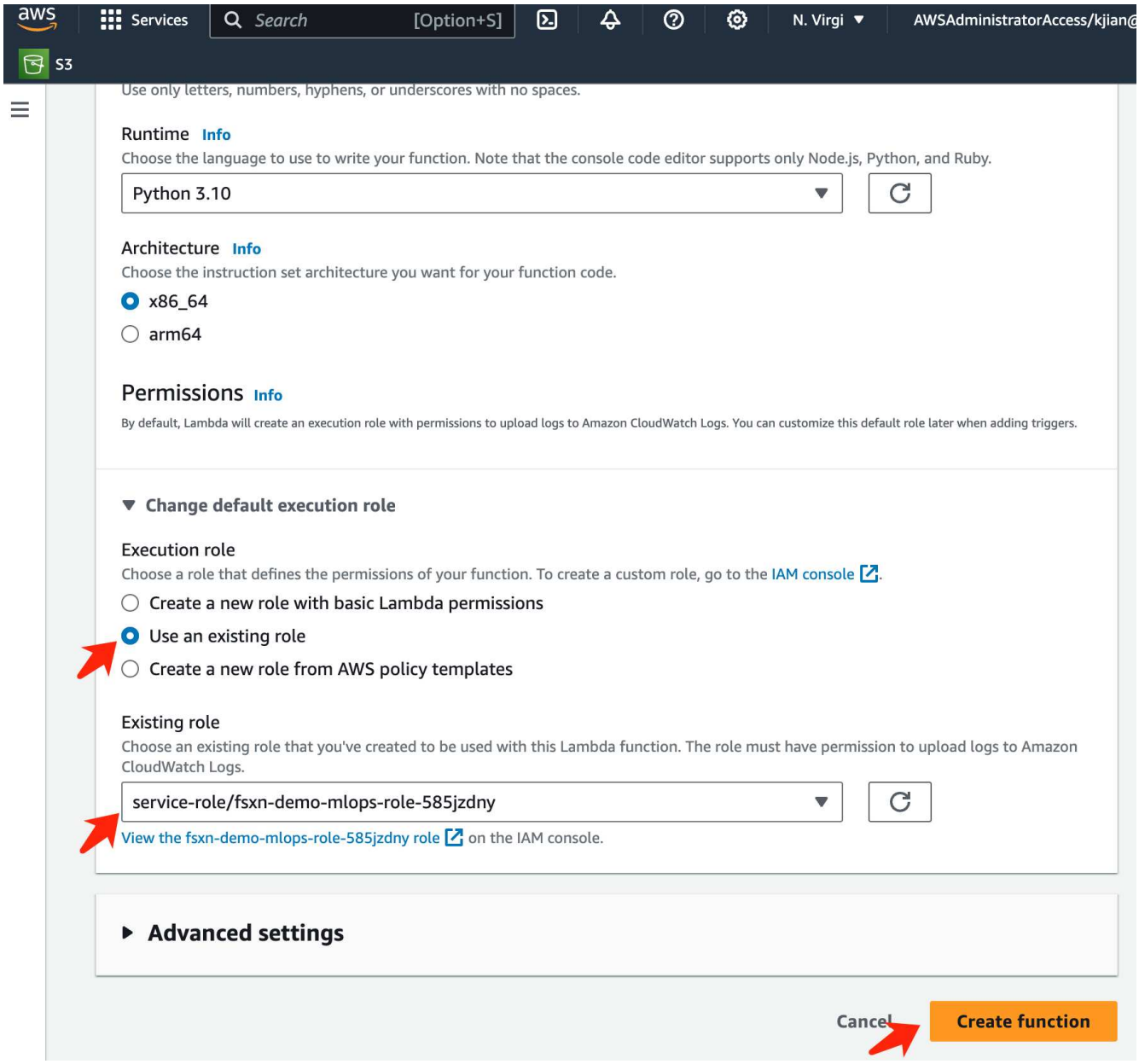
1. Para criar uma **função AWS Lambda**, navegue até o painel respectivo, alterne para a guia **Funções** e clique em **Criar função**.



2. Preencha todas as entradas necessárias na página e lembre-se de mudar o tempo de execução para **Python 3.10**.



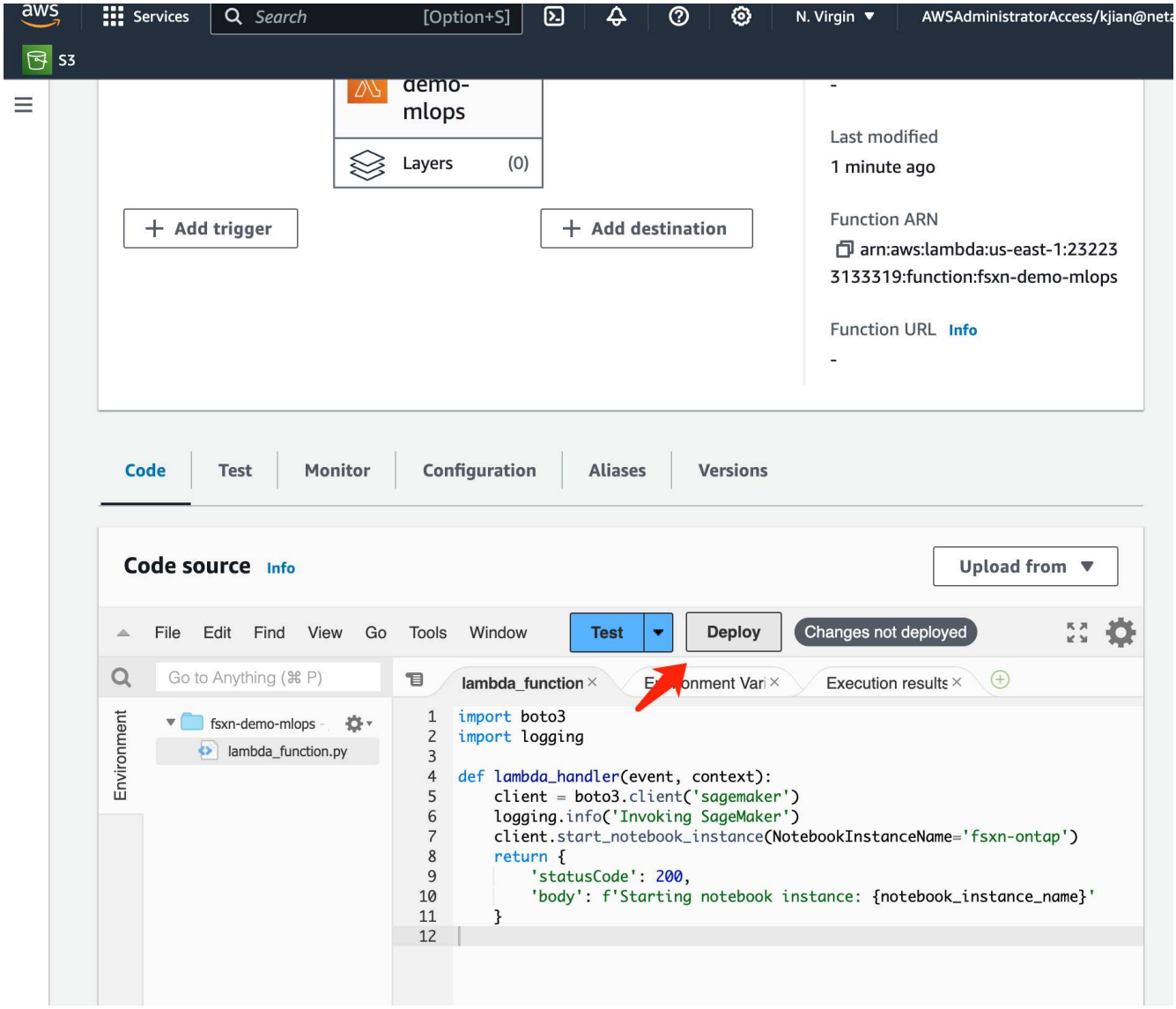
3. Verifique se a função designada tem a permissão necessária **AmazonSageMakerFullAccess** e clique no botão **Criar função**.



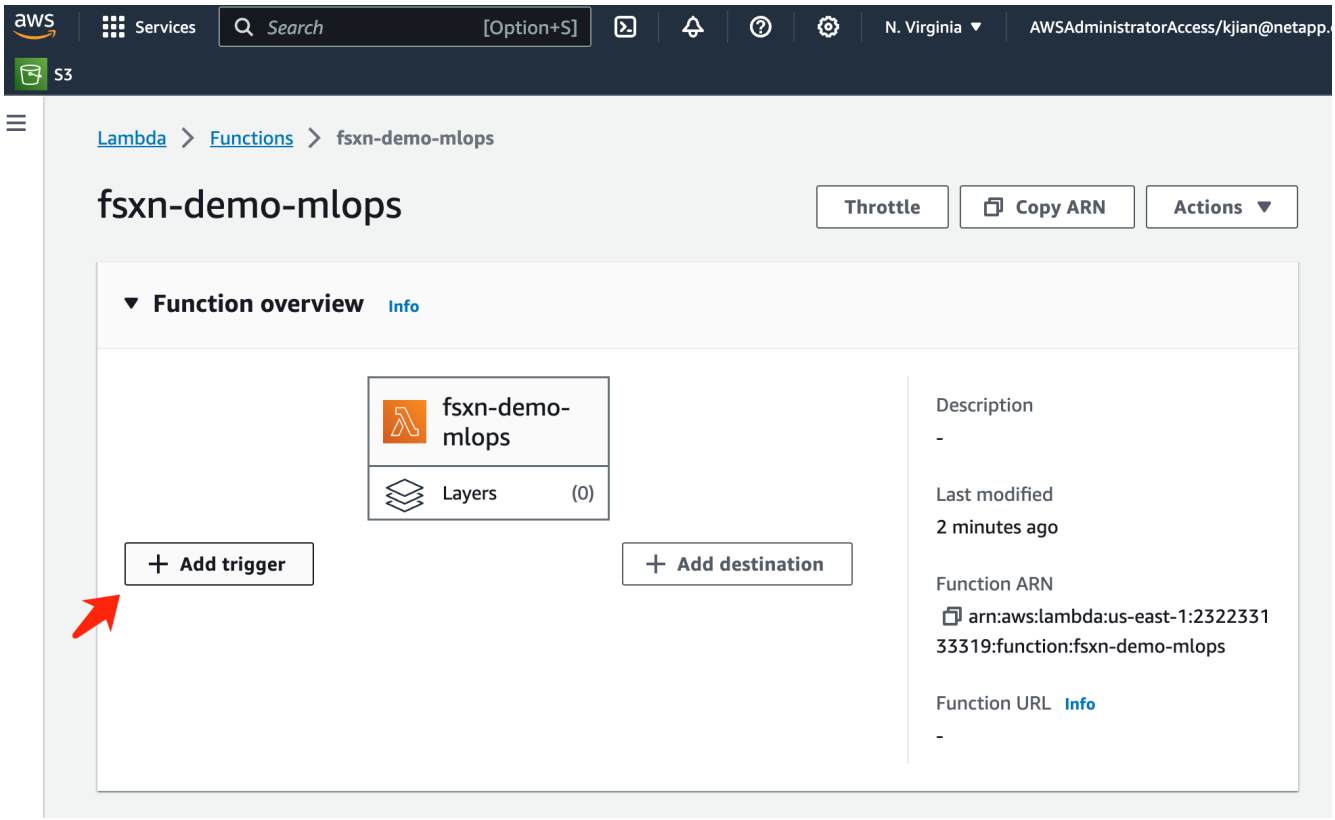
4. Selecione a função Lambda criada. Na aba de código, copie e cole o seguinte código na área de texto. Este código inicia a instância do notebook chamada **fsxn-ontap**.

```
import boto3
import logging

def lambda_handler(event, context):
    client = boto3.client('sagemaker')
    logging.info('Invoking SageMaker')
    client.start_notebook_instance(NotebookInstanceName='fsxn-ontap')
    return {
        'statusCode': 200,
        'body': f'Starting notebook instance: {notebook_instance_name}'
    }
```

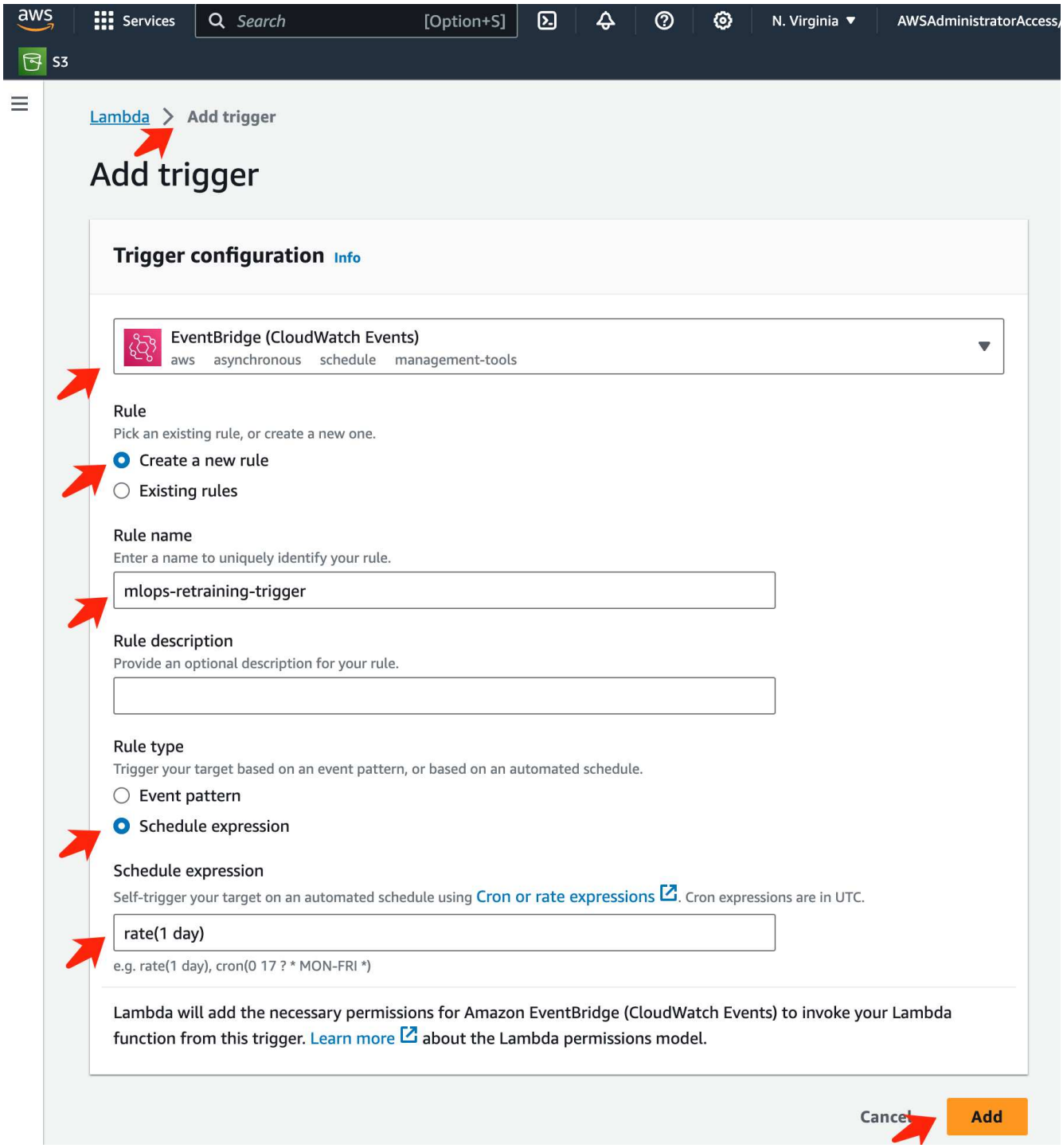
5. Clique no botão **Implantar** para aplicar esta alteração de código.



6. Para especificar como acionar esta função do AWS Lambda, clique no botão Adicionar acionador.



7. Selecione EventBridge no menu suspenso e clique no botão de opção Criar uma nova regra. No campo de expressão do cronograma, insira `rate(1 day)` e clique no botão Adicionar para criar e aplicar esta nova regra de tarefa cron à função do AWS Lambda.



Após concluir a configuração em duas etapas, diariamente, a **função AWS Lambda** iniciará o **SageMaker Notebook**, executará o retreinamento do modelo usando os dados do repositório **FSx ONTAP**, reimplantará o modelo atualizado no ambiente de produção e desligará automaticamente a **instância do SageMaker Notebook** para otimizar custos. Isso garante que o modelo permaneça atualizado.

Isso conclui o tutorial para desenvolver um pipeline de MLOps.

Informações sobre direitos autorais

Informações sobre marcas comerciais